Кафедра Прикладной математики
Института информационных технологий
РТУ МИРЭА

Дисциплина «Большие данные»

2022-2023 у.г.

Лекция 8. Технологии обработки больших объемов данных

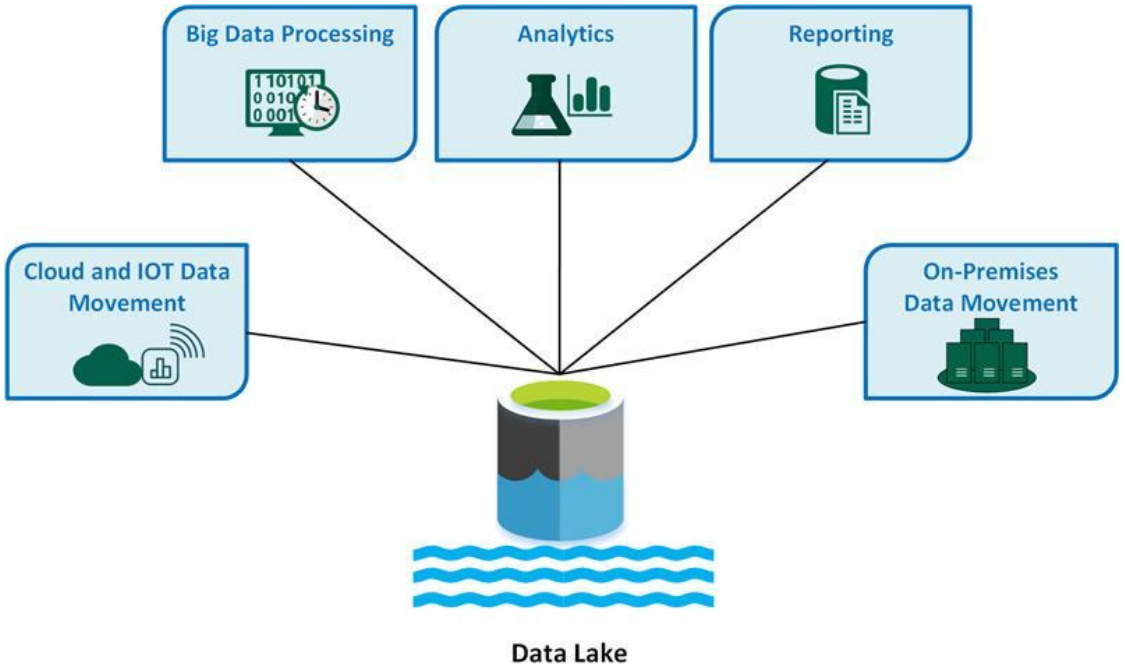
Часть 1. Озёра данных

Data Lake (озеро данных)

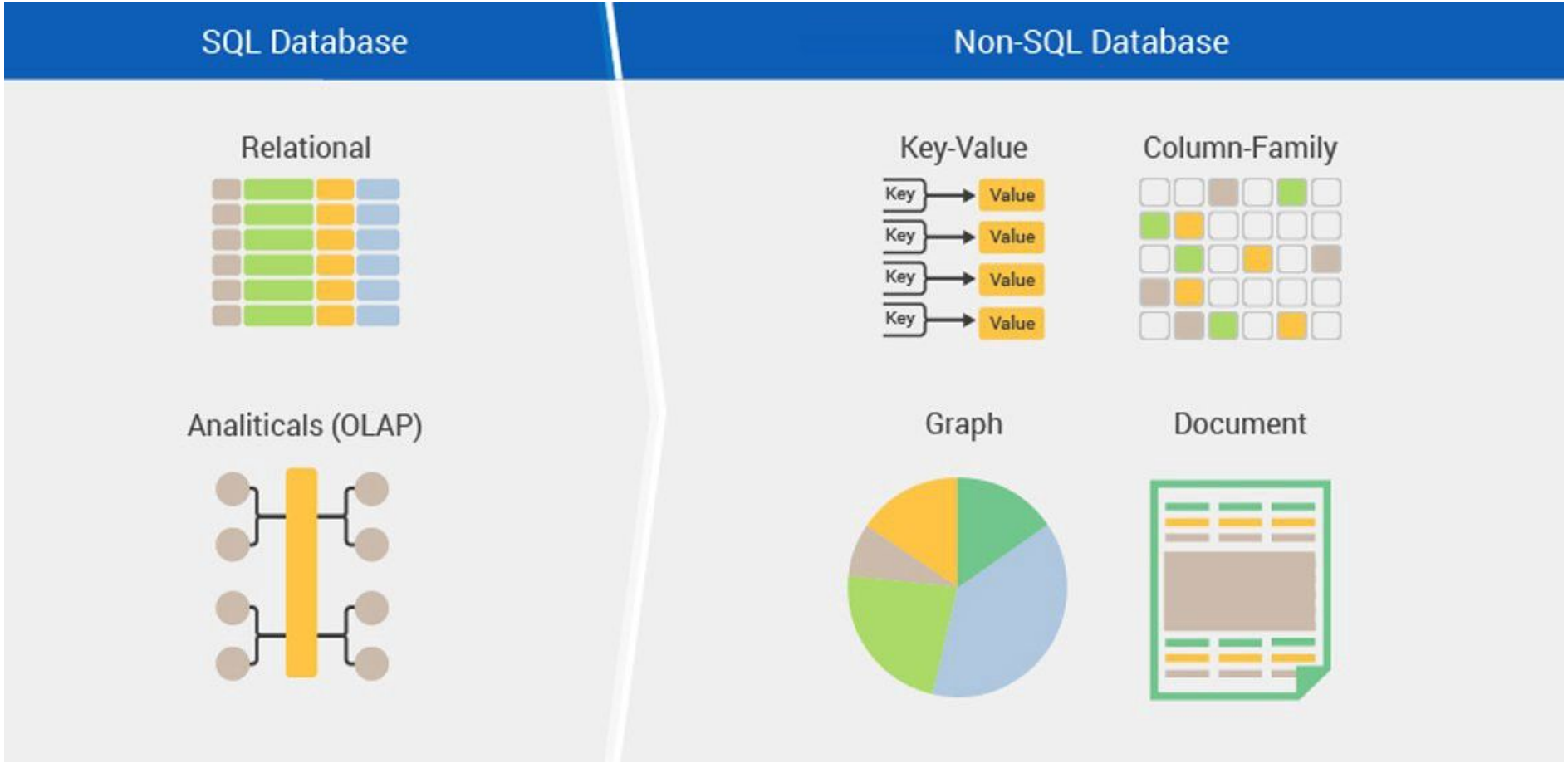
Data Lake — это репозиторий для хранения, который может вмещать большой объем данных в необработанном формате в виде файлов.

Хранилища Data Lake оптимизированы для масштабирования до **нескольких терабайт** и даже **петабайт данных**.

Данные обычно поступают из **нескольких разнородных источников** и могут быть **структурированными**, **частично структурированными** и **неструктурированными**.



Data Lake (озеро данных)

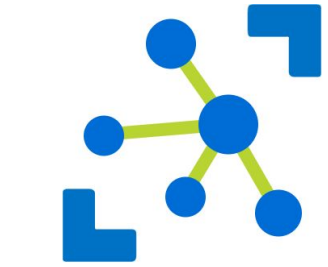


Идея и варианты использования

Идея, лежащая в основе Data Lake, — *хранение всех данных в исходном состоянии без каких-либо преобразований*. Такой подход отличает Data Lake от традиционного хранилища данных, в котором данные преобразуются и обрабатываются во время приема.

Основные варианты использования озера данных.

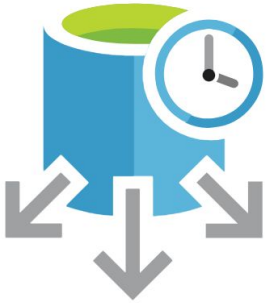
- Перемещение данных в облаке и IoT
- Обработка больших данных
- Аналитика
- Отчеты
- Перемещение локальных данных



Интернет вещей (IOT)

Аналитика данных (Data Analytics)

Отчеты



Перемещение локальных данных

Обработка больших данных

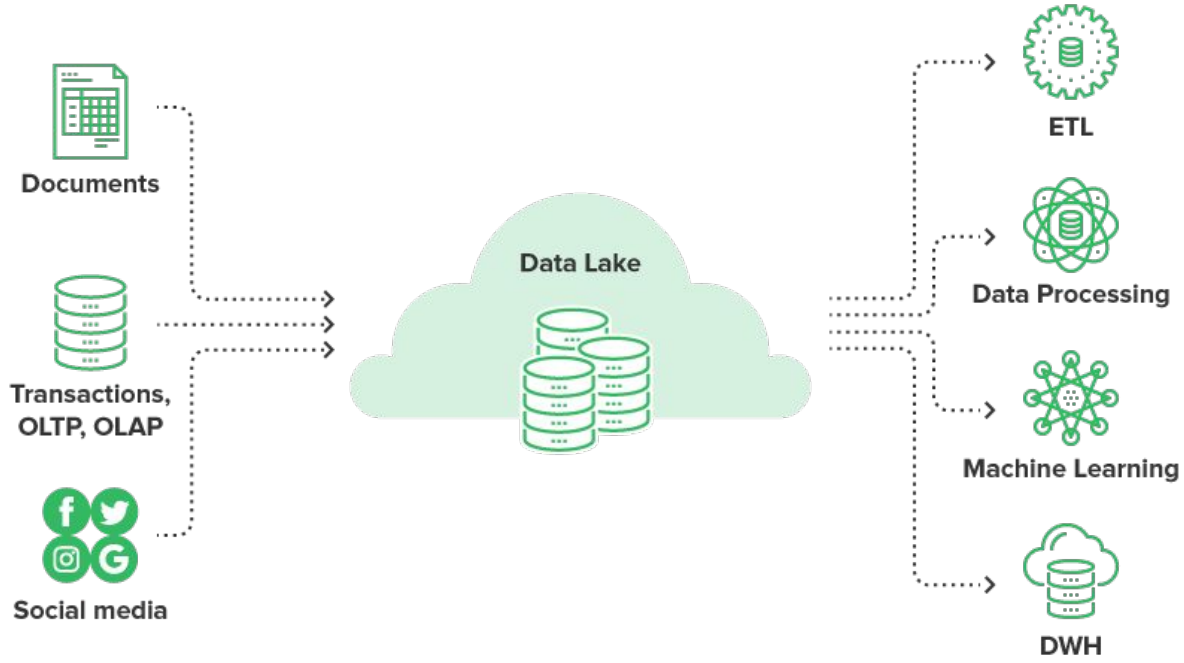
Преимущества Data Lake

Данные никогда не отклоняются от загрузки в хранилище, так как хранятся в необработанном формате. Это особенно полезно в окружении с большими данными, если заранее неизвестно, какие именно сведения будут получены в результате анализа данных.

Пользователи могут просматривать данные и создавать собственные запросы.

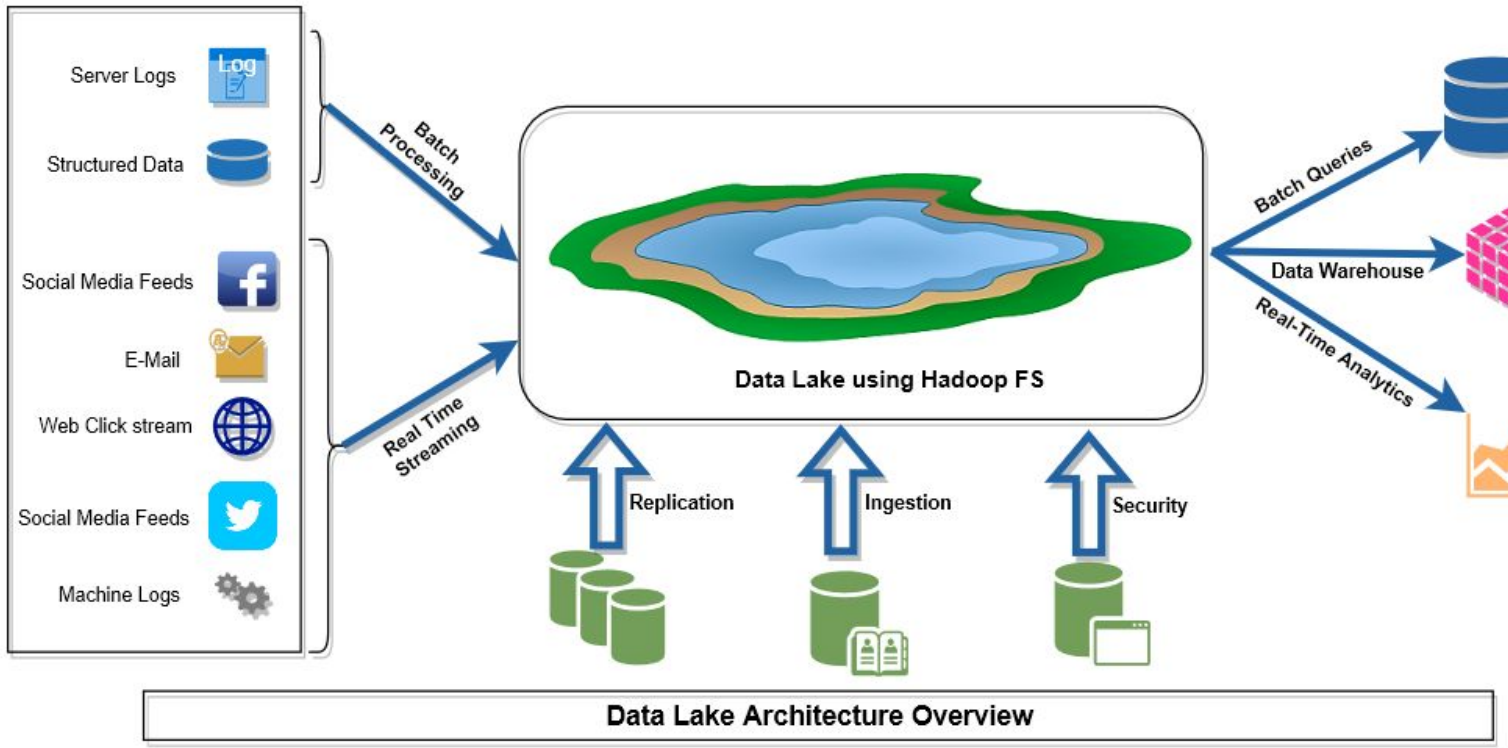
Может работать быстрее, чем традиционные средства ETL.

Обладает большей гибкостью, чем хранилище данных, так как дает возможность хранить частично структурированные и неструктурированные данные.



Данные DataLake

- Данные потока кликов
- Логи сервера
- Социальные сети
- Координаты геолокаций
- Данные с датчиков и устройств
- Структурированные данные
- Электронная почта

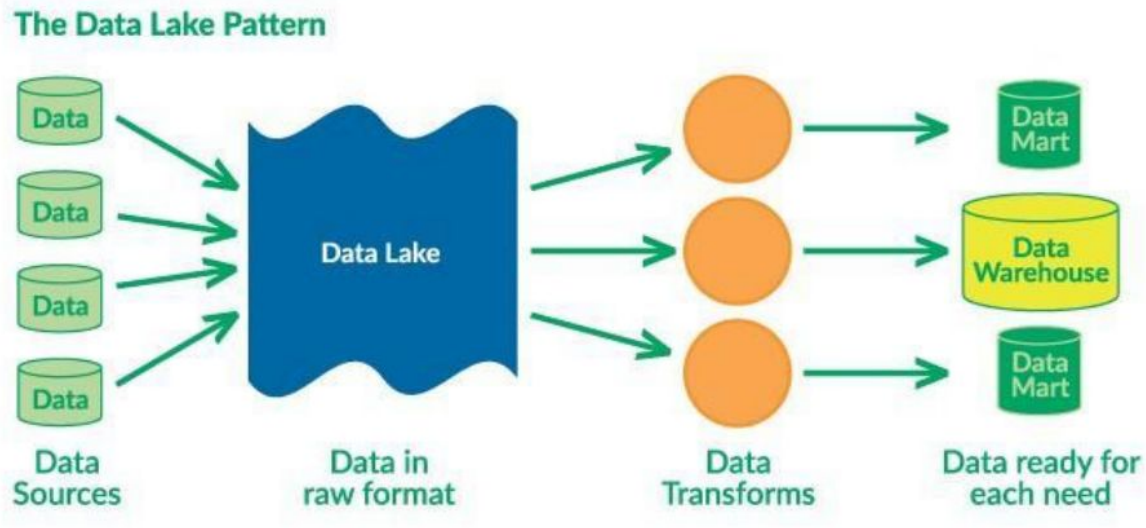


Data Lake

Полное решение Data Lake состоит из компонентов хранения и обработки данных.

Хранилище Data Lake создано для обеспечения отказоустойчивости, бесконечной масштабируемости и высокой пропускной способности при получении данных любых форм и размеров.

Компонент обработки Data Lake включает в себя один или несколько модулей обработки, созданных для этих целей, и может работать с данными, хранящимися в Data Lake в нужном масштабе.

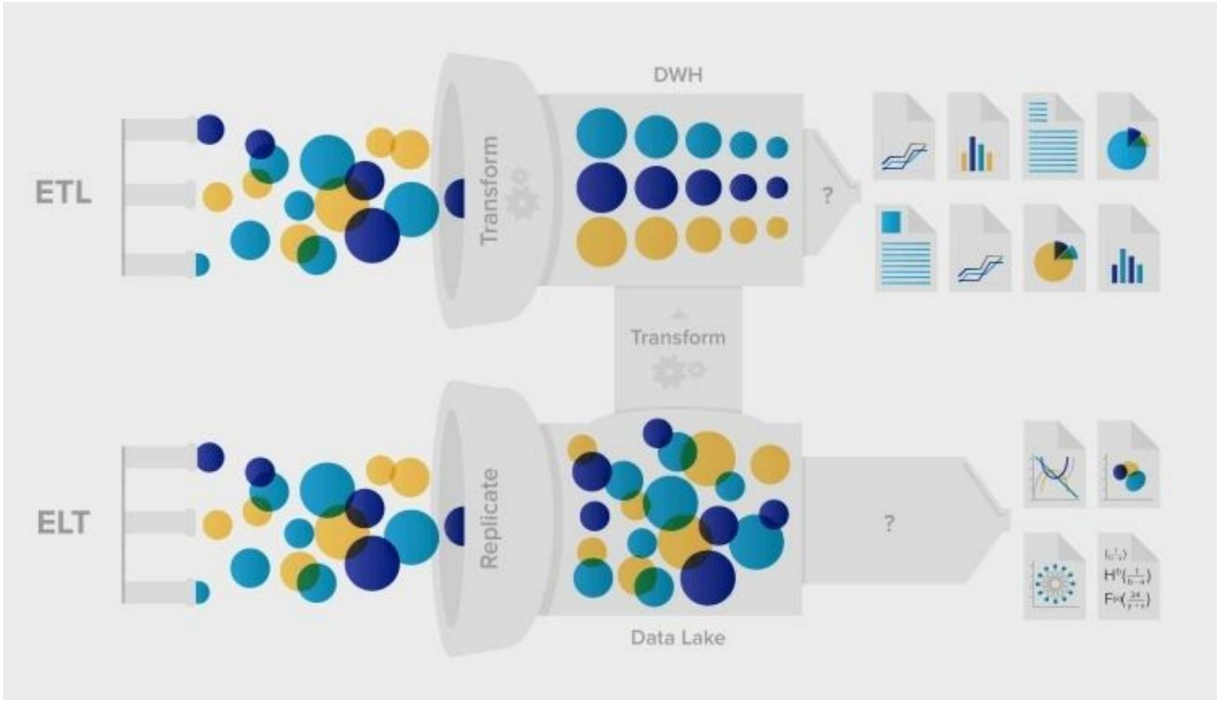


Когда следует использовать Data Lake

К наиболее распространенным сферам применения Data Lake относятся исследования данных, анализ данных и машинное обучение.

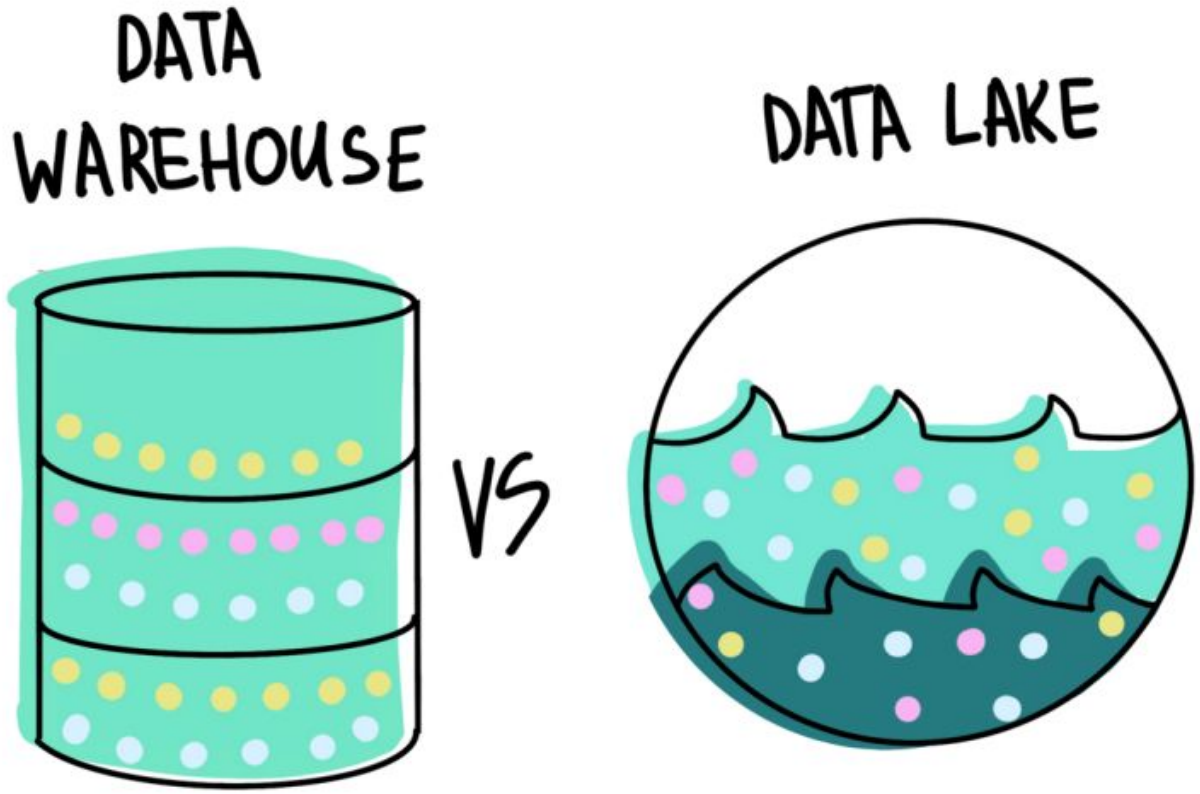
Data Lake также может служить источником данных для хранилища данных. При таком подходе необработанные данные поступают в Data Lake, а затем преобразуются в структурированный формат, поддерживающий запросы.

Хранилища Data Lake Store часто используются при потоковой передаче событий или в сценариях Интернета вещей, так как они могут хранить большие объемы реляционных и нереляционных данных без преобразования или определения схемы.



Сравнение с DWH

- Нагрузка
- Схема
- Масштабирование
- Методы доступа
- Преимущества
- Кто пользователи?
- SQL
- Данные



Сравнение с хранилищем данных

Область	Data Lake	Хранилище данных
Хранение данных	Может собирать и хранить неструктурированные, полу- и структурированные данные в виде файлов. Хранит все типы данных (в том числе и медиаформата). Не критичен тип источника и структура.	Собирает и хранит только структурированные данные. Хранилище данных хранит количественные метрики и их атрибуты. Данные перед загрузкой преобразуются и очищаются.
Определение схемы	Схема данных определяется уже после загрузки данных в Data Lake программистами при их чтении.	Схема данных определяется прямо перед хранением. Необходимо проделать работу по разработке схемы и базы данных. Предлагает взамен безопасность, производительность и унифицированную интеграцию.
Качество данных	Любые данные могут храниться, даже в сыром виде	Очень высокие требования к качеству данных. Таблицы должны быть нормализованы и очищены.

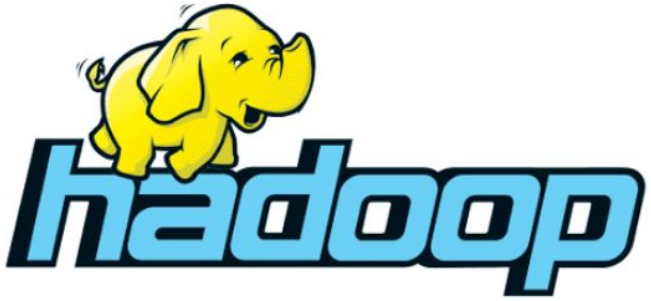
Сравнение с хранилищем данных

Область	Data Lake	Хранилище данных
Пользователи	Озеро данных идеально для пользователей, которые используют хранилище данных для глубокого анализа. Аналитики данных, Инженеры данных, Инженеры машинного обучения	Хранилище данных идеально для пользователей операционных данных, например бизнес-аналитики. Данные просто понимать, использовать и обрабатывать
Цена и производительность	Хранение данных является очень экономным ввиду алгоритмов сжатия и архивации данных. Запросы к данным получаются быстрыми в исполнении, так как информация хранится плотно с метаданными.	Стоимость хранения высока и запросы выполняются дольше ввиду хранения данных в разных таблицах (OLTP).
Доступность	Озера данных имеет несколько ограничений и данные легки в доступности.	Хранилище данных создано для хранения структурированных данных. Доступ к данным определяется навыками разработчика.

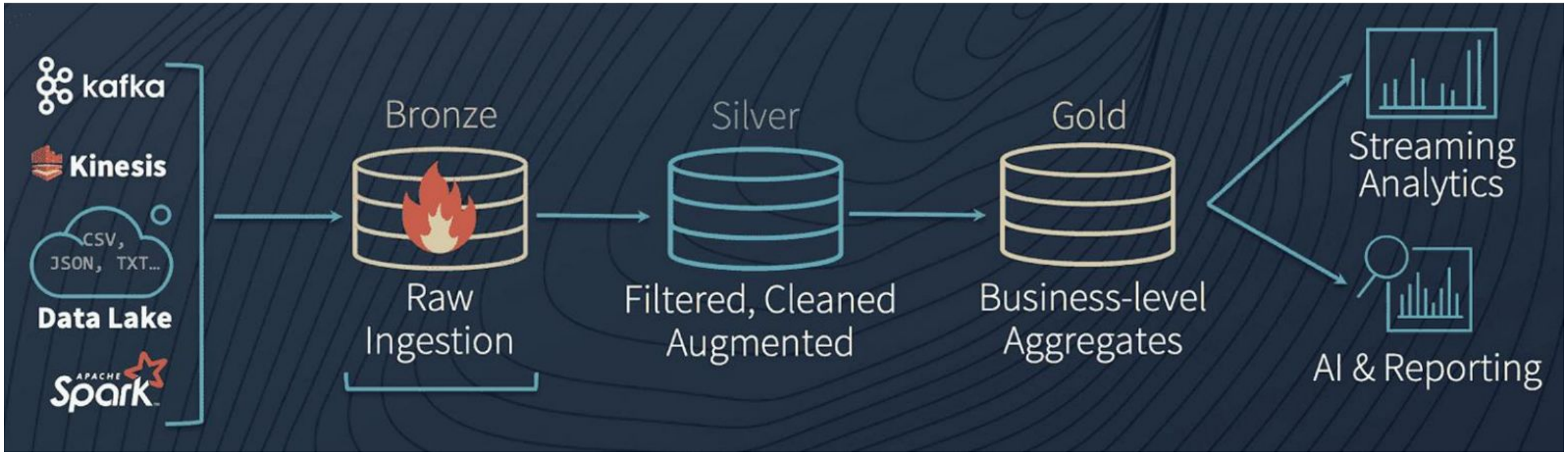
Сложности

- Отсутствие схемы и описательных метаданных создает трудности при использовании данных и создании запросов.
- Отсутствие семантической согласованности между данными может затруднять анализ данных, если пользователи не обладают профессиональными навыками в этой области.
- Качество данных, поступающих в Data Lake, сложно гарантировать.
- Без надлежащего управления могут возникать проблемы с контролем доступа и конфиденциальностью. Какие данные поступают в Data Lake, кто может их использовать и с какой целью?
- Data Lake может оказаться не лучшим способом интеграции данных, которые уже являются реляционными.
- Само по себе хранилище Data Lake не поддерживает интегрированный или целостный просмотр данных для всей организации.
- Data Lake может превратиться в "свалку" данных, которые никогда не будут использоваться для изучения и анализа.

Инструменты

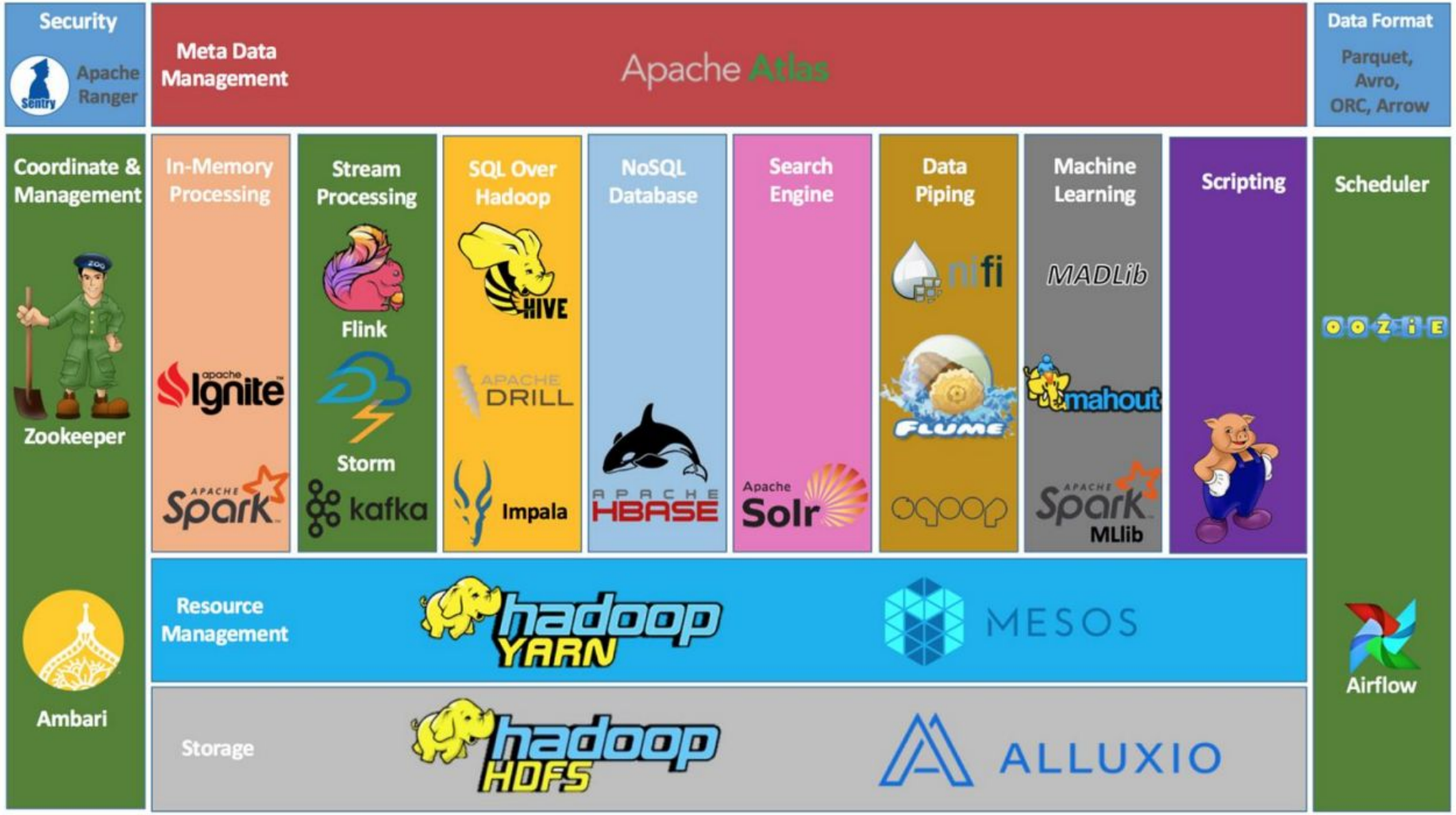


Подготовка данных к работе



Часть 2. Экосистема Hadoop. Файловая система HDFS. Обработка данных с применением MapReduce

Экосистема Hadoop



Элементы Hadoop

Hadoop – программный комплекс для хранения и обработки больших объемов слабоструктурированной информации, состоящий из:

- Подсистемы хранения распределенной файловой системы Hadoop – **HDFS**
- Подсистемы автоматического управления ресурсами кластера для балансировки нагрузки – **Yarn**
- Подсистемы пакетной обработки данных с применением дублирования (отображения) и агрегации – **MapReduce**

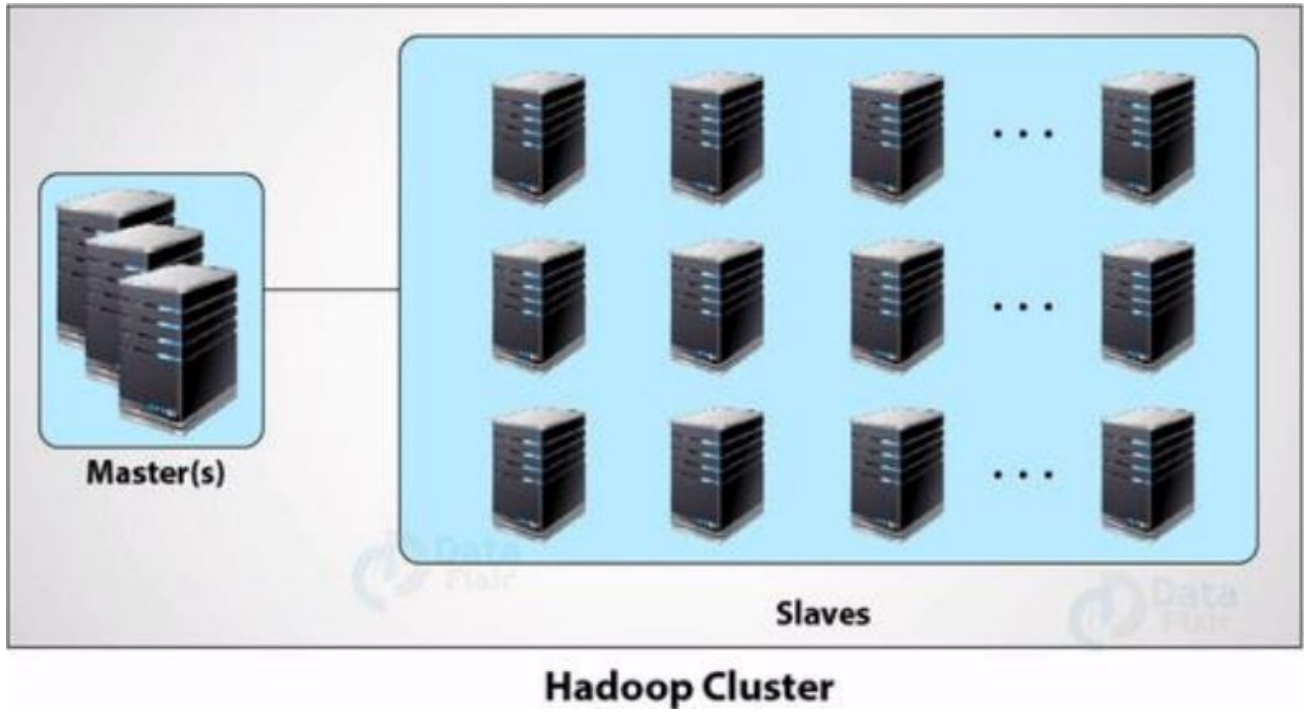
Система Hadoop

Hadoop обслуживает распределенный кластер на программном уровне, эмулируя файловую систему Linux.

Hadoop работает на основе Master/Slave архитектуры в двух возможных режимах, невысокой и высокой доступности.

В режиме высокой доступности Hadoop решает проблему единой точки отказа главного узла.

Работа кластера Hadoop организуется за счет служб NameNode, SecondaryNameNode, DataNode.



Hadoop Cluster

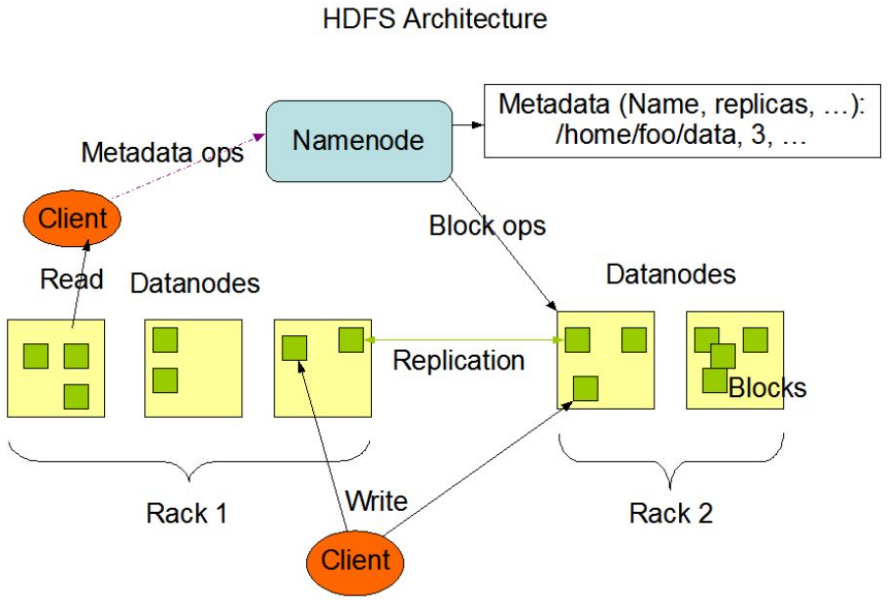
Распределенная файловая система

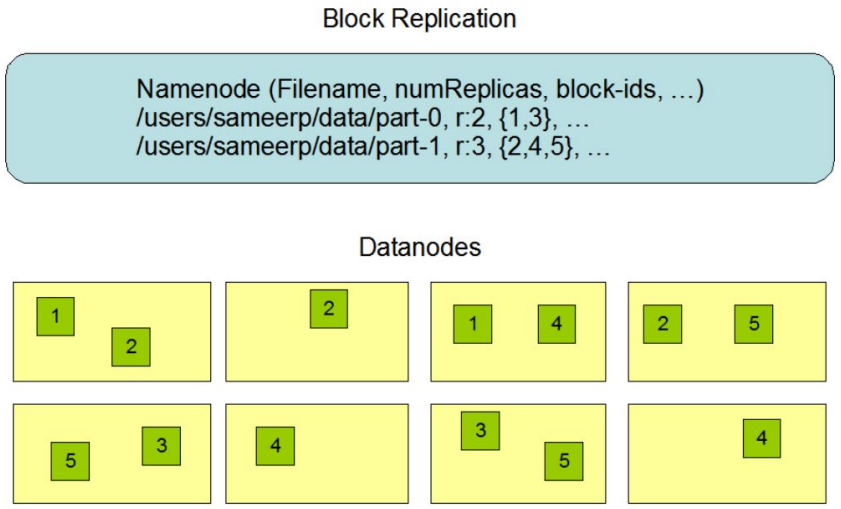
Подсистема хранения **HDFS** разбивает файлы на блоки фиксированного размера. Размер блока при хранении файлов 64/128 Мб.

Файл хранится в подчиненных узлах **DataNode**. Репликация блоков происходит на ближайших узлах.

Информация о блоках файлов хранится в главном узле **NameNode.**

Все управление распределенной файловой системой происходит через **NameNode.** Демон (служба) NameNode объединяет виртуально в себе весь ресурс хранения данных, в то время как он связан по сети.





Система пакетной обработки данных

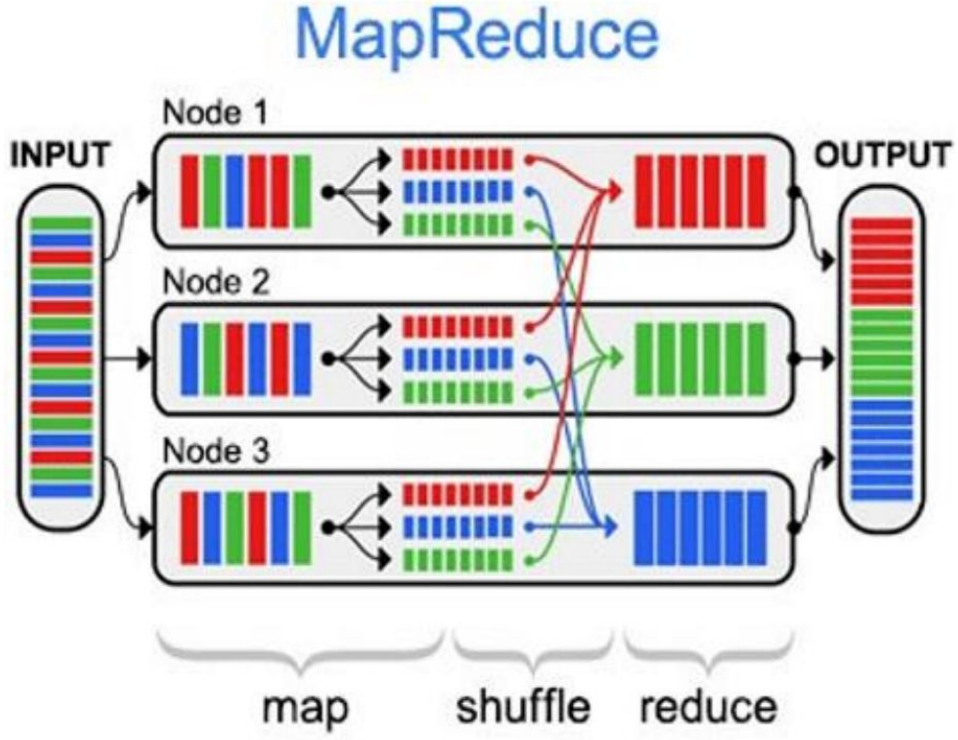
MapReduce обрабатывает данные ключ:значение по принципу:

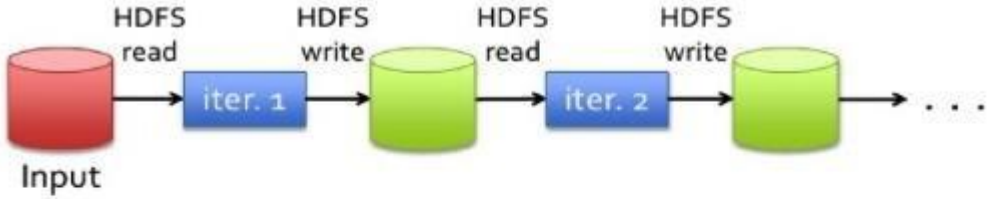
1. Применения к значениям операций или функций на стадии **Map**,
2. Перемешивания ключей между исполнителями в распределенной системе хранения на стадии **Shuffle**
3. Агрегации информации в блоках данных по принципу объединения данных по ключу на стадии **Reduce**

Процедура основана на репликации блоков данных между узлами данных.

Данные на этапе Map обрабатываются на местах и происходит запись на диск.

После перемешивания агрегации происходят уже на других исполнителях.





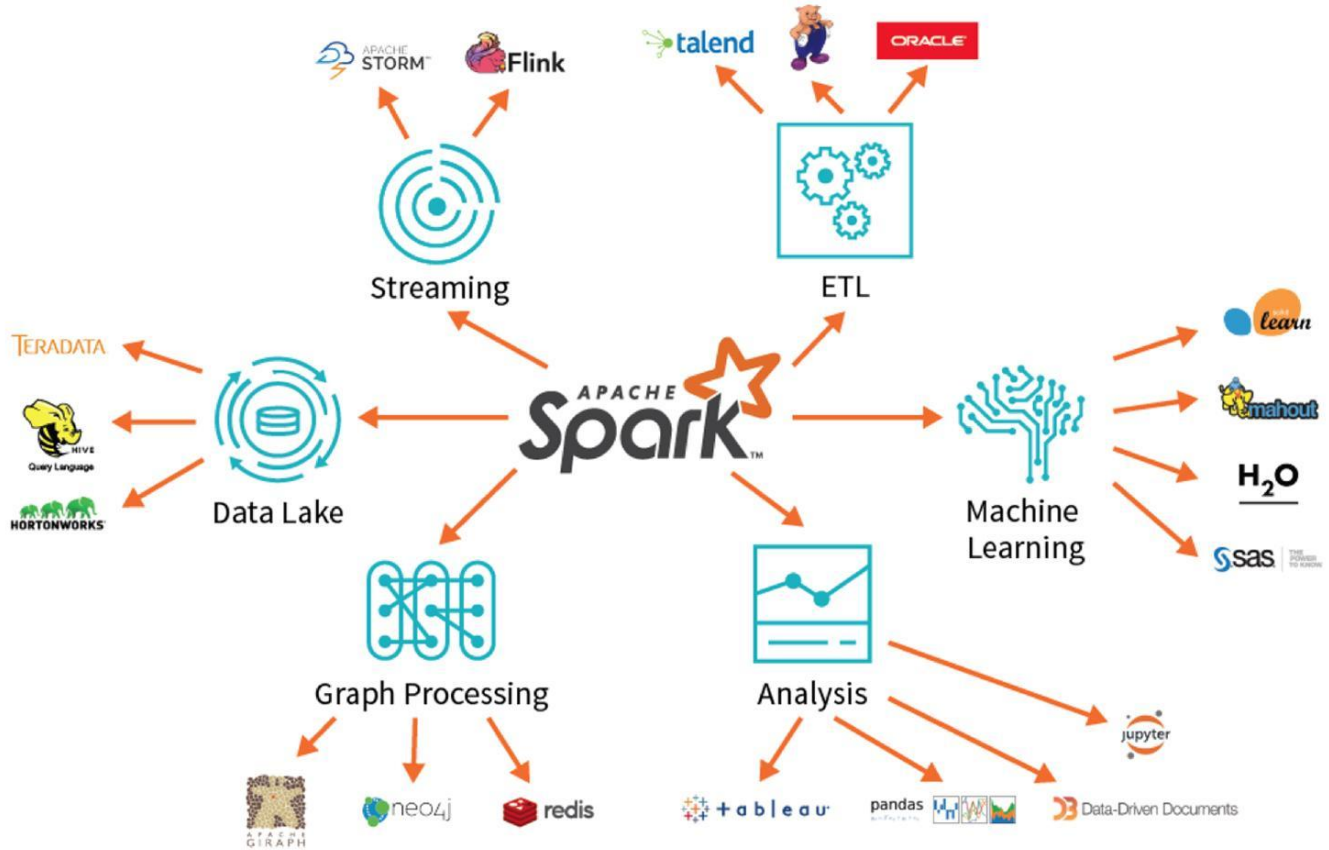
Spark для обработки в памяти

Apache Spark – система пакетной обработки данных в памяти.

В отличие от библиотек Hadoop, которые осуществляют обработку данных на основе чтения/записи в диск, обработка данных происходит в энергозависимой памяти.

Spark обладает большей производительностью в сравнении с Hadoop, но меньшую отказоустойчивость.

Spark также обеспечивает полный цикл обработки и анализа больших данных за счет обширной экосистемы и библиотек.

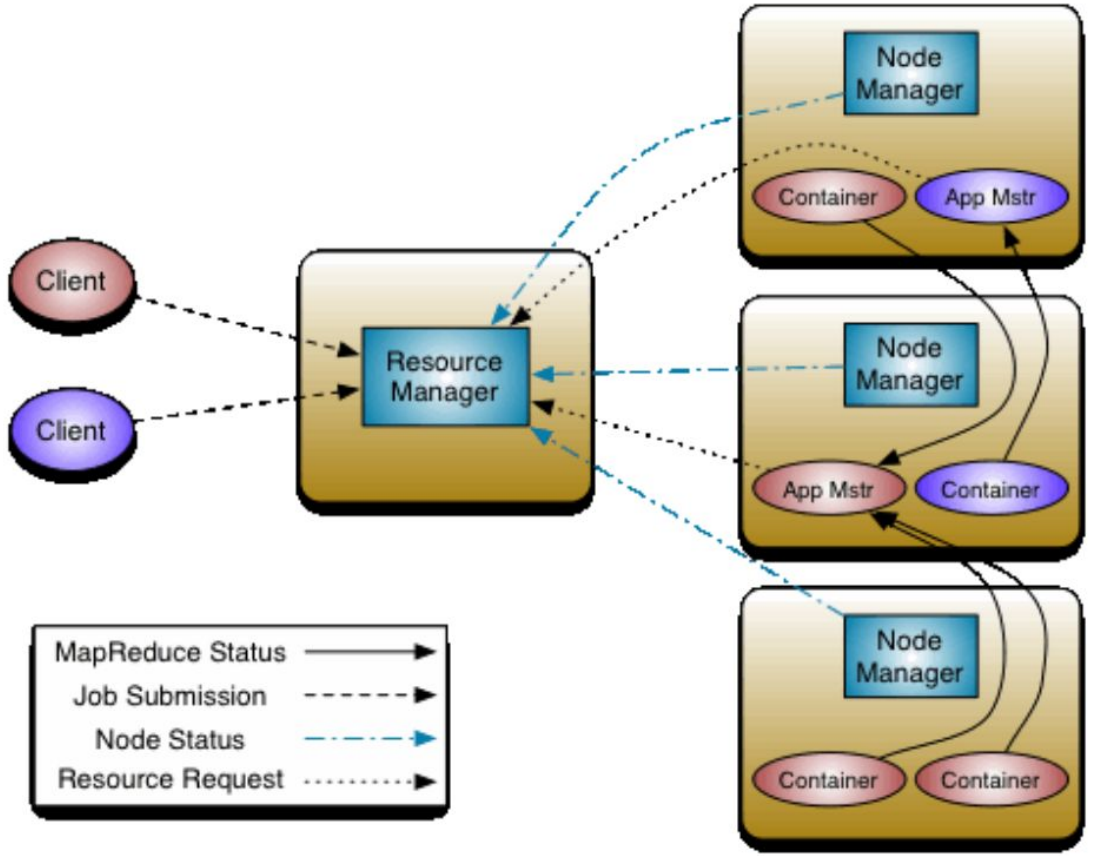


Система управления ресурсами кластера

YARN – это система планирования заданий и управления кластером (Yet Another Resource Negotiator), которую также называют

MapReduce 2.0 – набор системных программ (демонов), обеспечивающих совместное использование, масштабирование и надежность работы распределенных приложений.

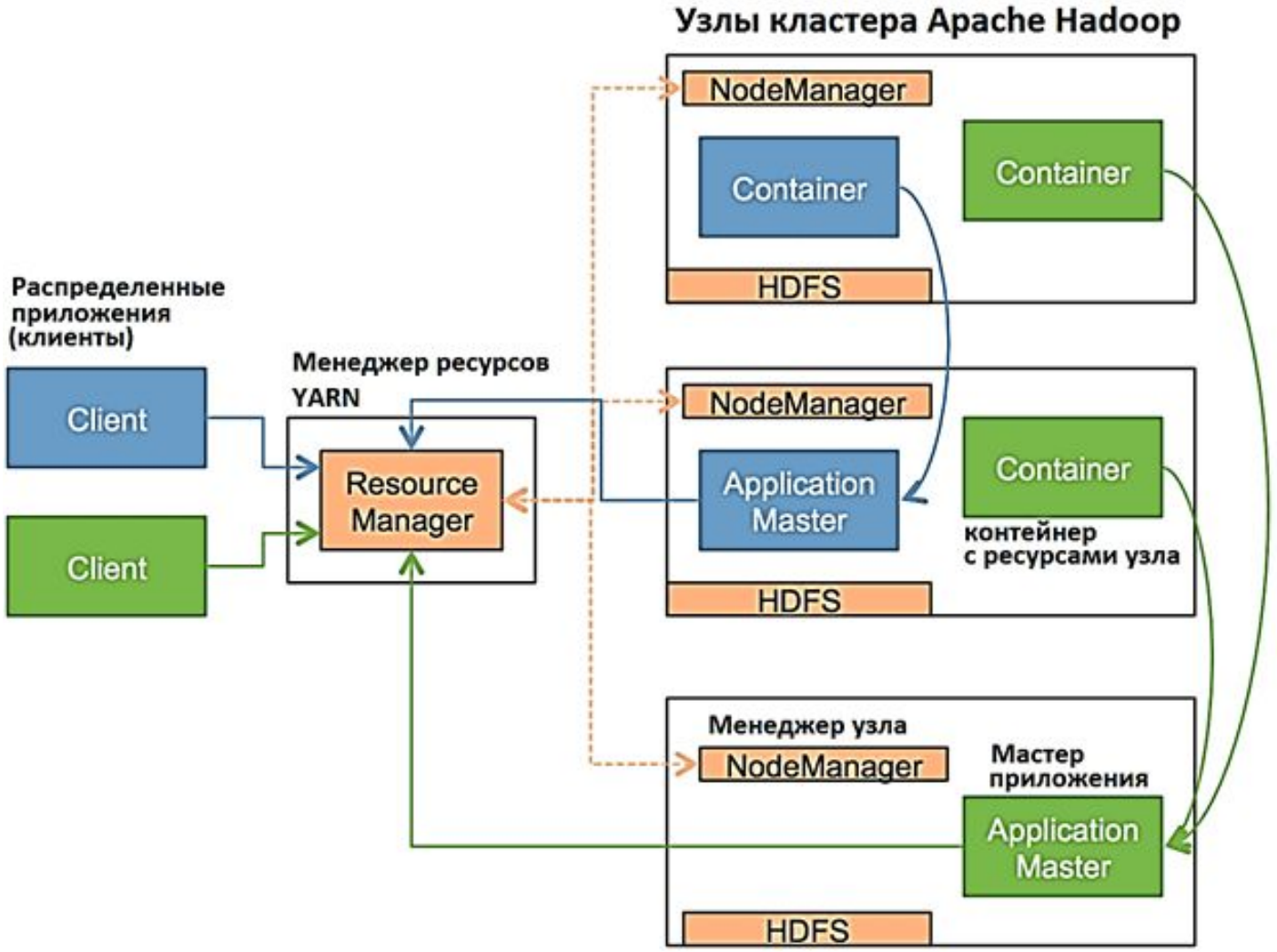
YARN является интерфейсом между аппаратными ресурсами кластера и приложениями, использующих его мощности для вычислений и аналитики больших данных.



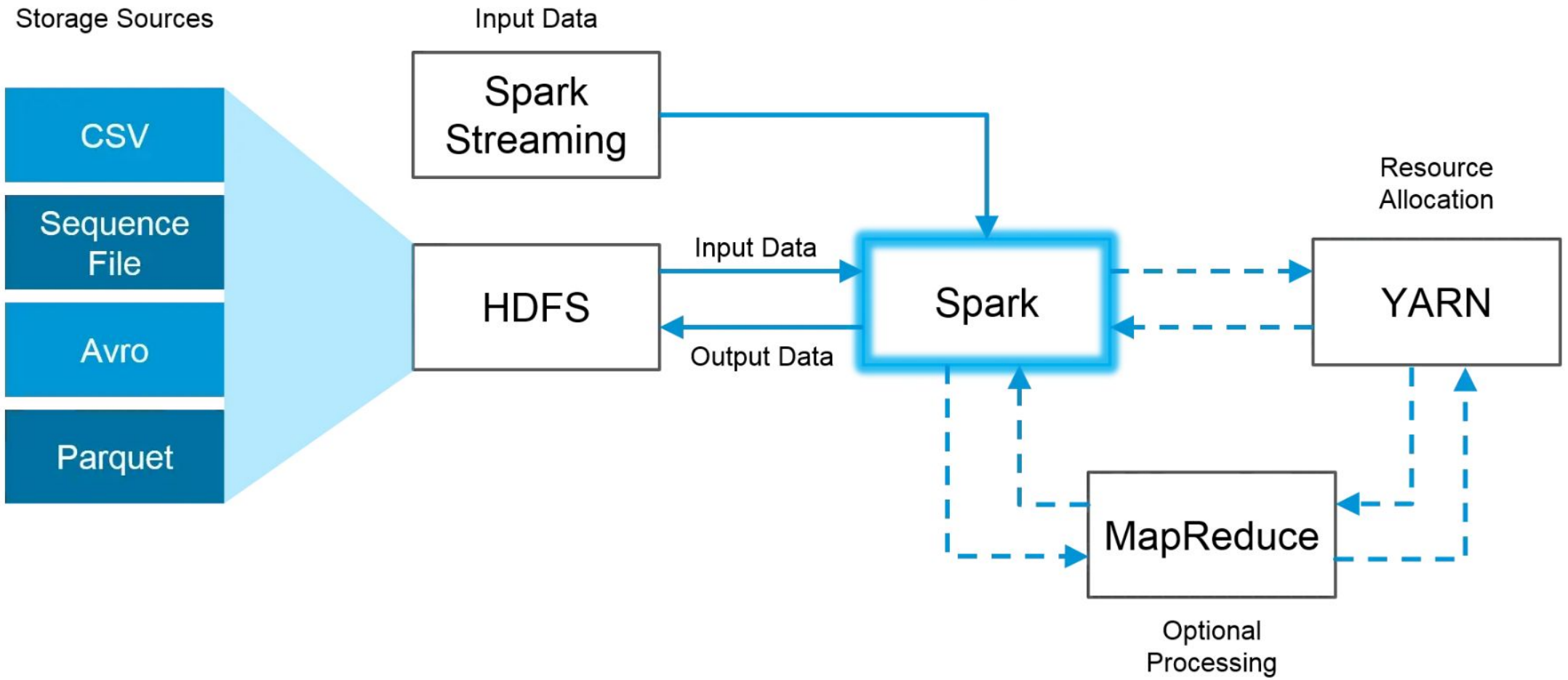
Система управления ресурсами кластера

- ***ResourceManager (RM)*** — менеджер ресурсов, которых отвечает за распределение ресурсов, необходимых для работы распределенных приложений, и наблюдение за узлами кластера, где эти приложения выполняются. ResourceManager включает планировщик ресурсов (Scheduler) и диспетчер приложений (ApplicationsManager, AsM).
- ***ApplicationMaster (AM)*** – мастер приложения, ответственный за планирование его жизненного цикла, координацию и отслеживание статуса выполнения, включая динамическое масштабирование потребления ресурсов, управление потоком выполнения, обработку ошибок и искажений вычислений, выполнение локальных оптимизаций. Каждое приложение имеет свой экземпляр ApplicationMaster. ApplicationMaster выполняет произвольный пользовательский код и может быть написан на любом языке программирования благодаря расширяемым протоколам связи с менеджером ресурсов и менеджером узлов.
- ***NodeManager (NM)*** – менеджер узла – агент, запущенный на узле кластера, который отвечает за отслеживание используемых вычислительных ресурсов (CPU, RAM и пр.), управление логами и отправку отчетов об использовании ресурсов планировщику. NodeManager управляет абстрактными контейнерами – ресурсами узла, доступными для конкретного приложения.
- ***Контейнер (Container)*** — набор физических ресурсов (ЦП, память, диск, сеть) в одном вычислительном узле кластера.

Принцип работы Hadoop YARN



Организация работы Hadoop



Часть 3. Потоки данных. Обмен данными в системах BigData

Пакетная обработка данных

Система пакетной обработки данных – конвейер обработки данных, состоящий из систем извлечения, обработки и хранения данных, где в качестве единицы обработки информации выделяют контейнер единообразной информации одной структуры.

Например, изменяют или отбирают нужные файлы по заданным критериям. Выбранное действие обязательно применяется сразу ко всем файлам/байтам/записям в пакете.

Существуют различные методы группировки данных по разным контейнерам-пакетам:

1. По времени создания. Например данные, поступившие за последние 30 минут.
2. По типу данных. Видеофайлы – в один контейнер, таблички по продажам в другой.
3. По источнику данных.
4. По содержимому.
5. Вручную по заданным критериям.

Отобранные данные отправляются в систему пакетной обработки данных, где с ними происходят нужные действия.

Обработка и применение

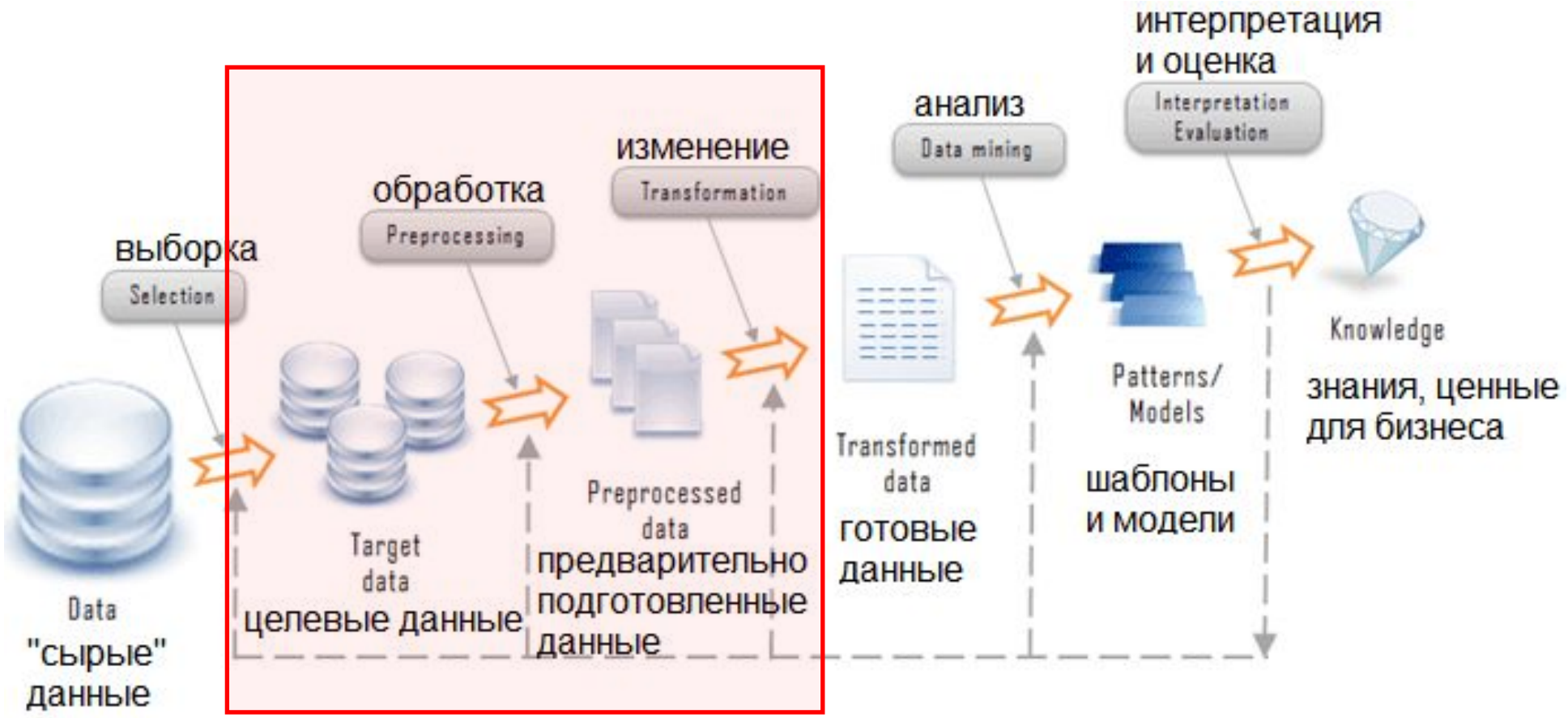
При пакетной обработке данных с данными в одном пакете может происходить:

- **Применение операций.** Выбранная операция применяется к каждому элементу пакета.
- **Фильтрация.** Можно фильтровать файлы внутри пакета — например, оставить в нем только картинки с котами и удалить все остальные. Или фильтровать пакет в целом — пропускать данные на дальнейшую обработку тогда, когда в нем встретились фотографии только котов.

Применение

- Для разделения сложных процессов на мелкие, понятные и легко реализуемые операции. Разбивка задач на мелкие подзадачи и применение этих подзадач к группам файлов отлично для этого подходит.
- Для того чтобы ускорить работу с данными. Пакетную обработку данных можно параллелить и запускать в кластерах серверов, то есть сразу на нескольких серверах.
- Комбинация обеих причин — сложные многоступенчатые вычисления на больших объемах данных.

Пакетная обработка данных



Пакетная обработка данных

Примеры применения пакетной обработки данных:

- обработка данных с применением MapReduce
- стандартная аналитика данных с применением аналитических платформ и языков программирования

Минусы пакетной обработки (batch):

- данные доставляются с задержкой.
- создаётся пиковая нагрузка на железо.

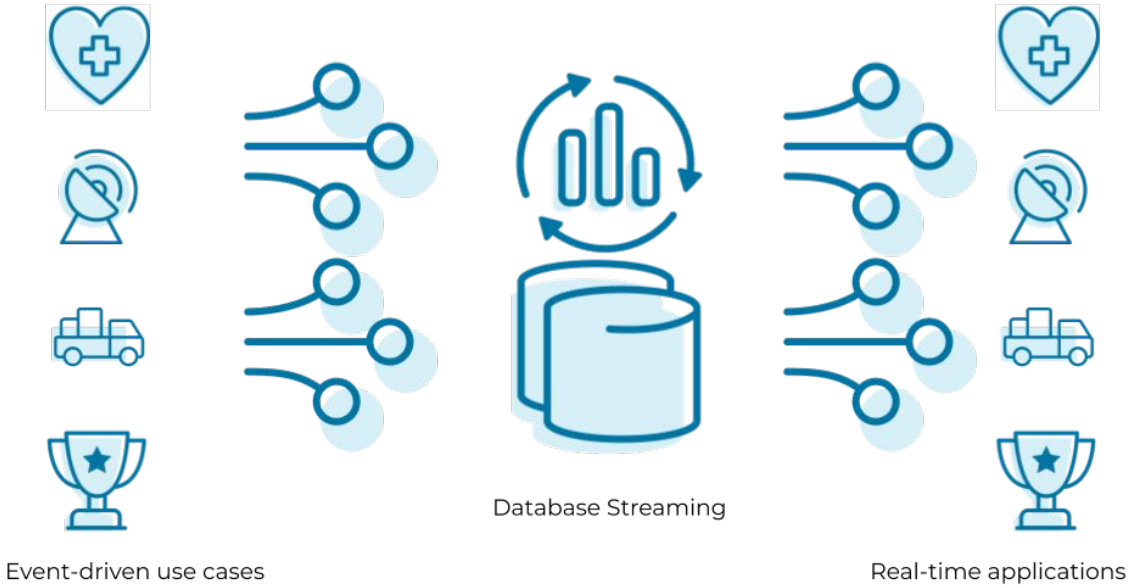
Но у пакетной обработки есть и плюсы:

- высокая эффективность.
- простота разработки и поддержки.

Потоковая обработка данных

Потоковая обработка это однопроходная парадигма обработки данных, которая всегда поддерживает данные в движении для достижения низкой задержки обработки.

Будучи более высокой абстракцией систем обмена сообщениями, потоковая обработка поддерживает не только агрегацию и доставку сообщений, но и способна выполнять асинхронные вычисления в реальном времени при передаче информации.



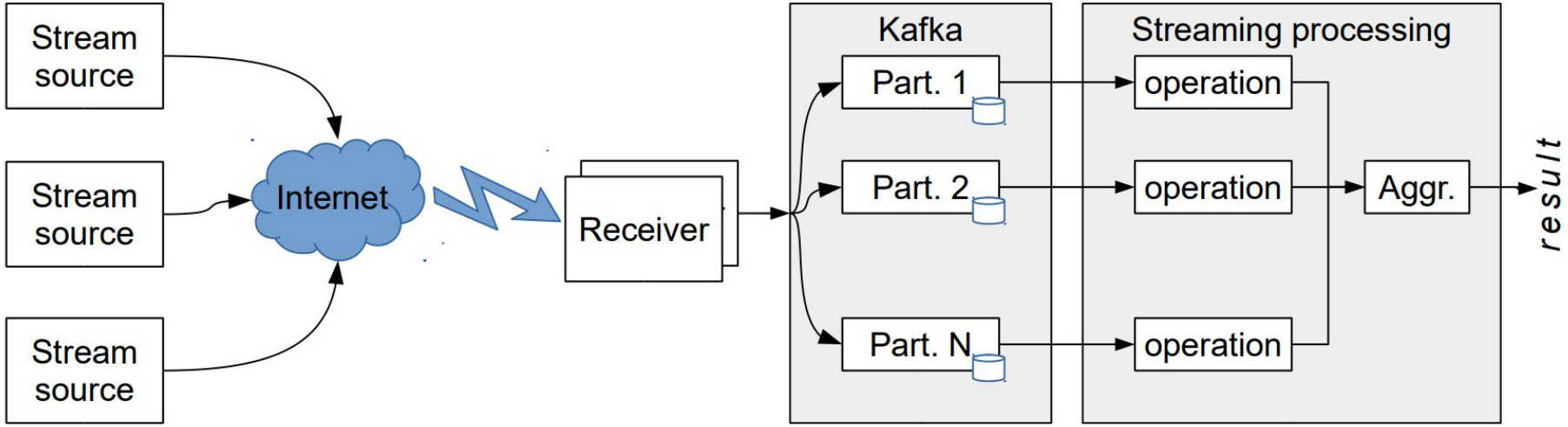
Потоковая обработка данных

Плюсы потоковой обработки данных (streaming):

- результат в режиме реального времени
- равномерная нагрузка на железо

Главный минус потоковой обработки:

- сложность разработки и поддержки.



Элементы потоковой обработки

Элементы системы потоковой обработки данных

1. **Загрузчик данных** (средство доставки данных до хранилища);

- Apache Flume или Apache NIFI, StreamSets
- **2. Шина обмена данными** (нужна не всегда, но в стримах без неё никак, т. к. вам потребуется система, через которую вы будете обмениваться данными в реал-тайме);
- ApacheKafka, RabbitMQ, NATS
- **3. Хранилище данных** (как же без него);
- Apache HDFS+Hive, Apache Kudu+Impala, Yandex ClickHouse
- **4. ETL-движок** (необходим, чтобы делать различные фильтрации, сортировки и прочие операции);
- **5. BI** (чтобы выводить результаты);
- **6. Оркестратор** (связывает весь процесс воедино, организовывая многоэтапную обработку данных).

Лямбда архитектура

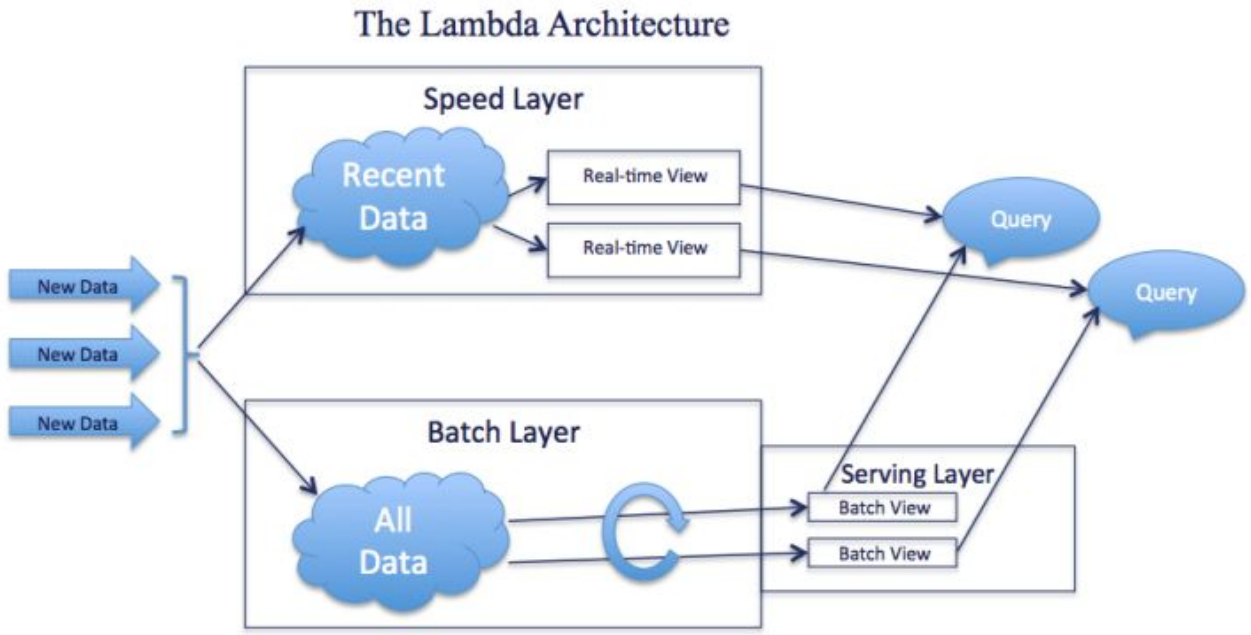
Лямбда архитектура поддерживает параллельную обработку пакетных данных и потоковых данных на основе параллельного исполнения служб в слое пакетной и скоростной обработки.

Преимущества:

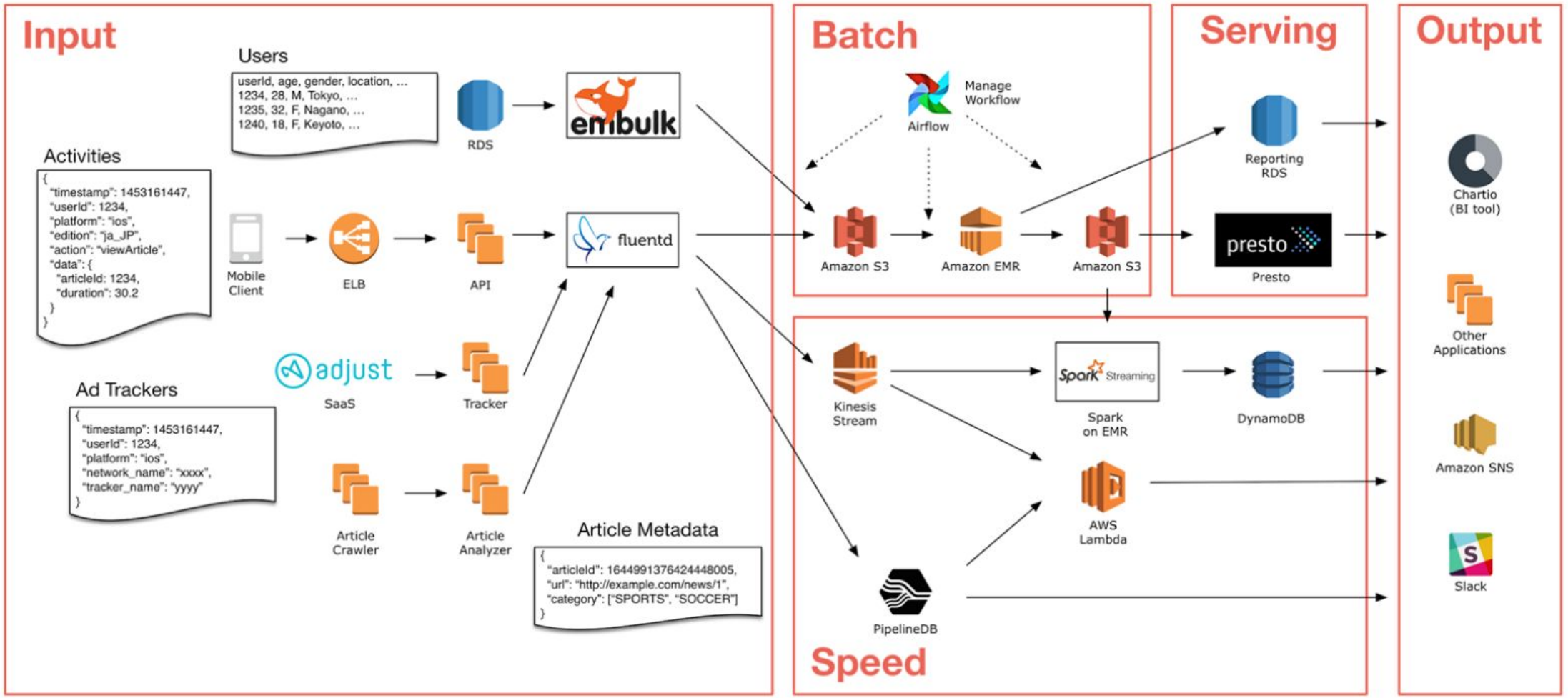
- Скорость обработки
- Простота построения ахритектуры
- Отсутствие слияния двух потоков

Недостатки:

- Высокая нагрузка на вычислительный кластер
- Большой объем данных для хранения
- Необходимость в запросах из двух потоков



Лямбда архитектура



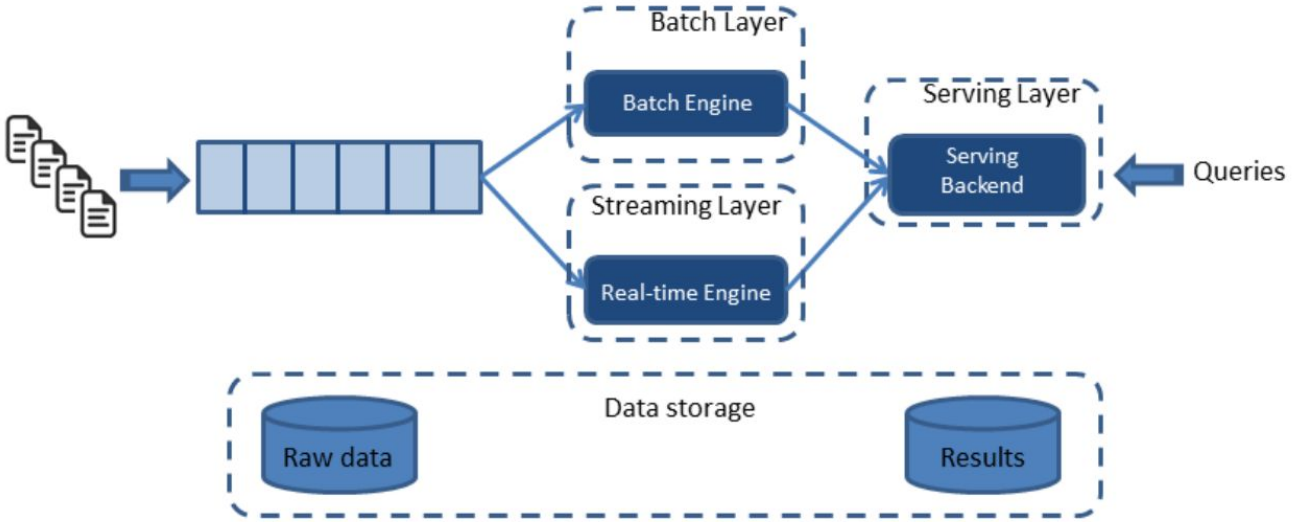
Каппа архитектура

Отличается от лямбда-архитектуры в слиянии направлений обработки данных: потоковой и пакетной.

Данные хранятся в одном хранилище данных.

Запросы можно производить к единому хранилищу данных.

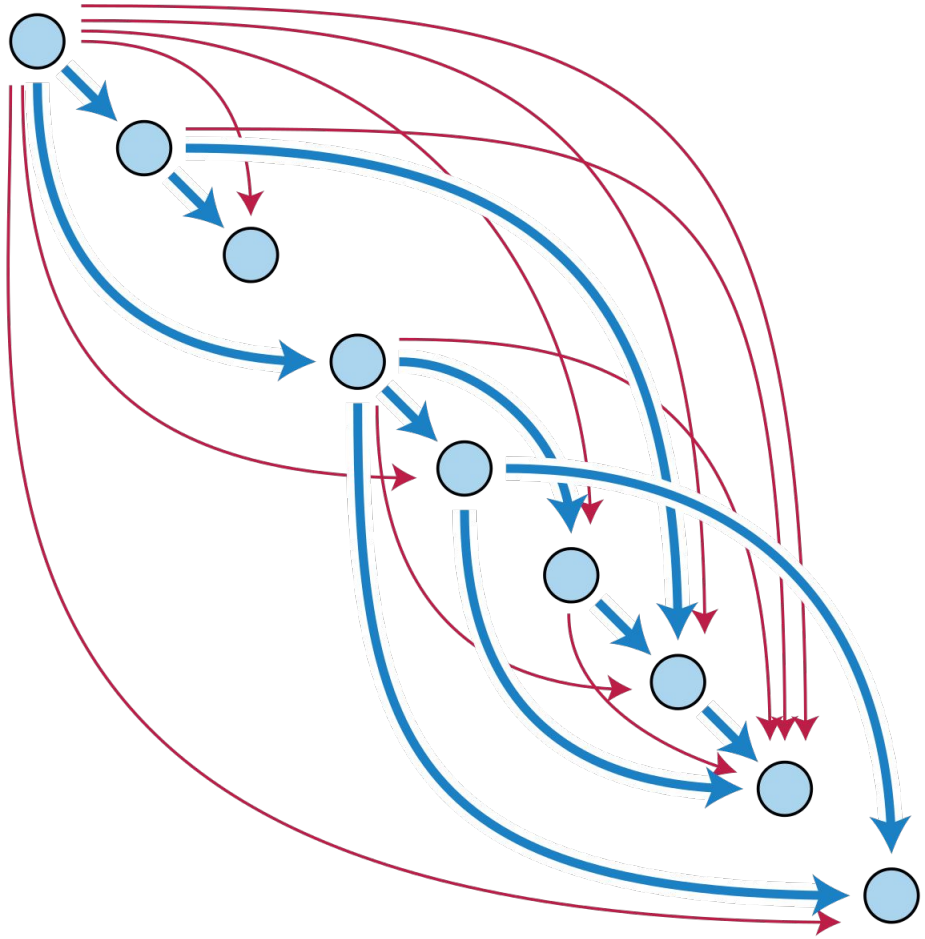
Для каппа-архитектуры результаты потоковой обработки не нужны в оперативных отчетностях, а только в виде диагностической информации в случае проблем.



Направленные ациклические графы

Направленный ациклический граф (DAG) представляет собой ориентированный граф без направленных циклов. То есть он состоит из вершин и ребер (также называемых дугами), каждое ребро которых направлено от одной вершины к другой, так что следование этим направлениям никогда не приведет к образованию замкнутого цикла.

Концепция приводит к нотации и инструментам для организации потоков данных между подключениями, миграциями и приложениями.

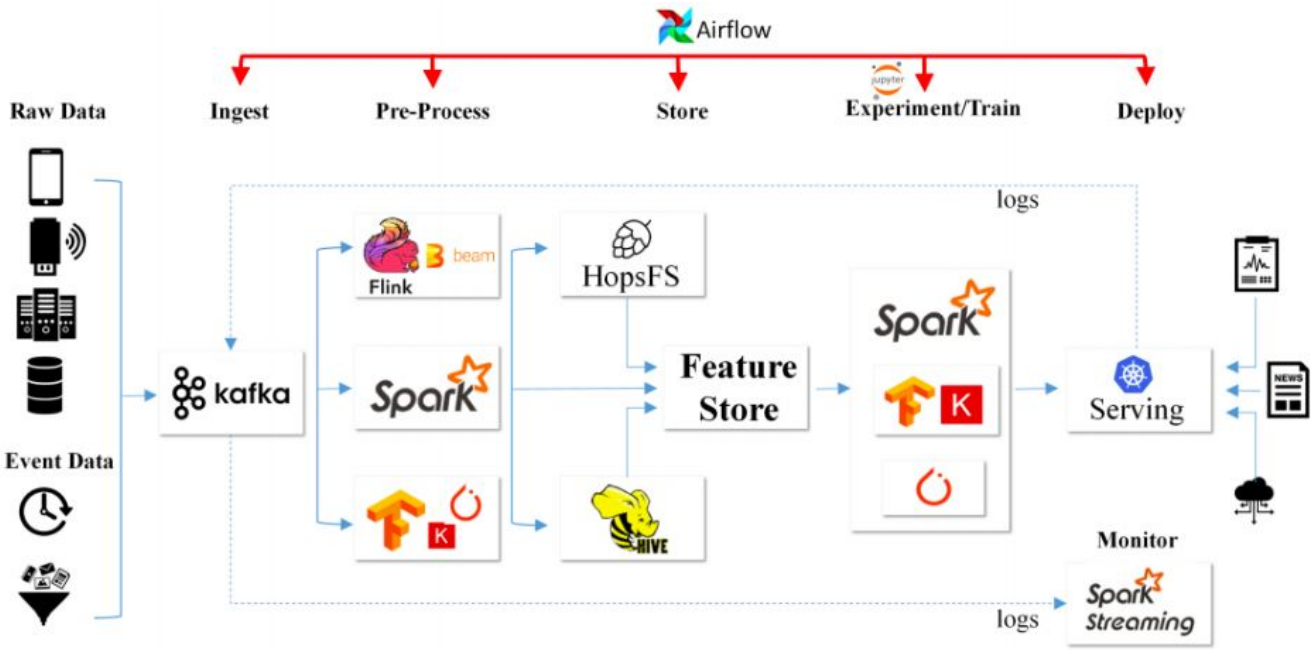


Apache Airflow

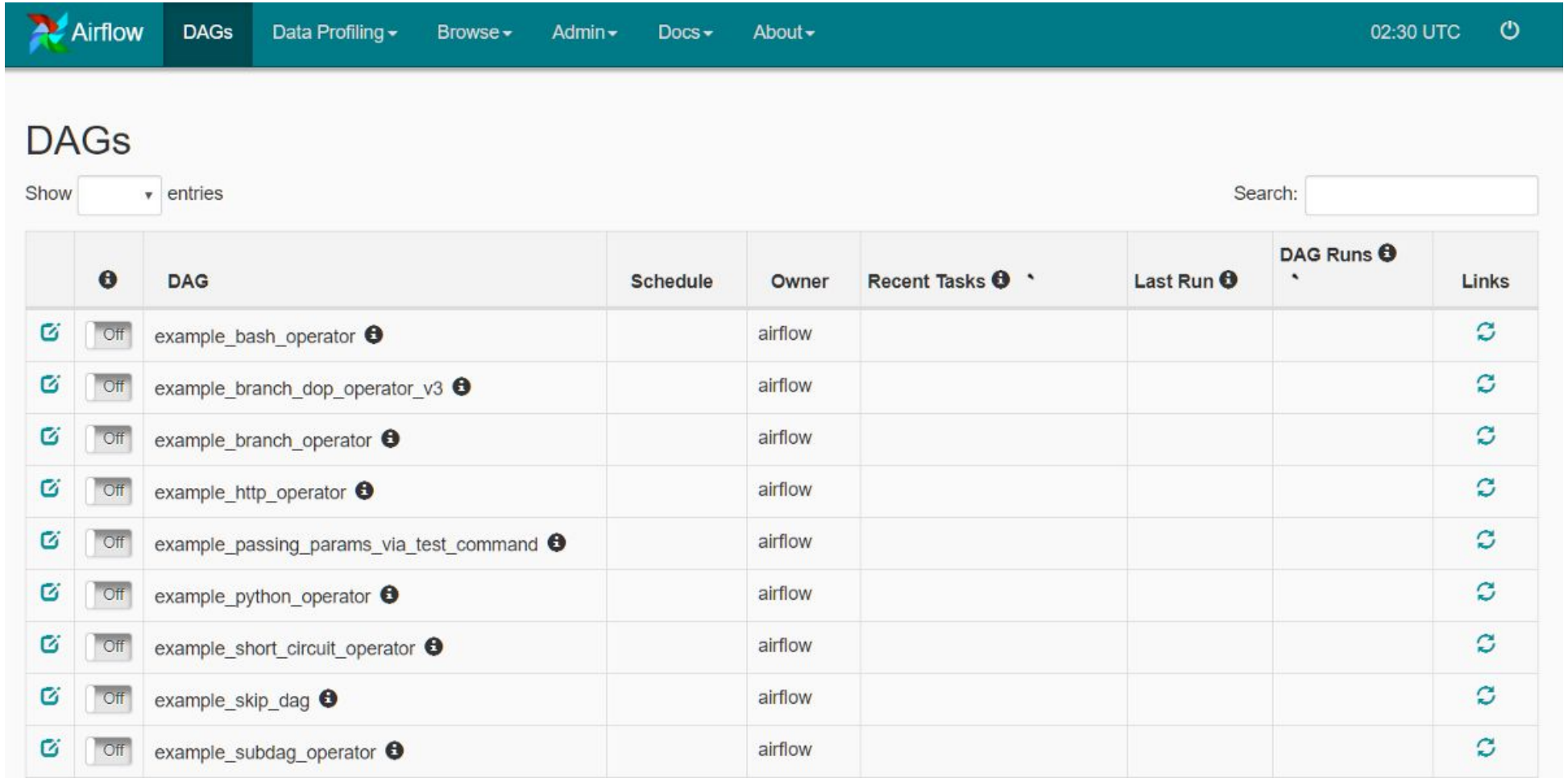
Apache Airflow — открытое программное обеспечение для создания, выполнения, мониторинга и оркестровки потоков операций по пакетной обработке данных.

Написан на **Python**, потоки операций и зависимости между ними кодируются также на Python по принципу «конфигурация как код»

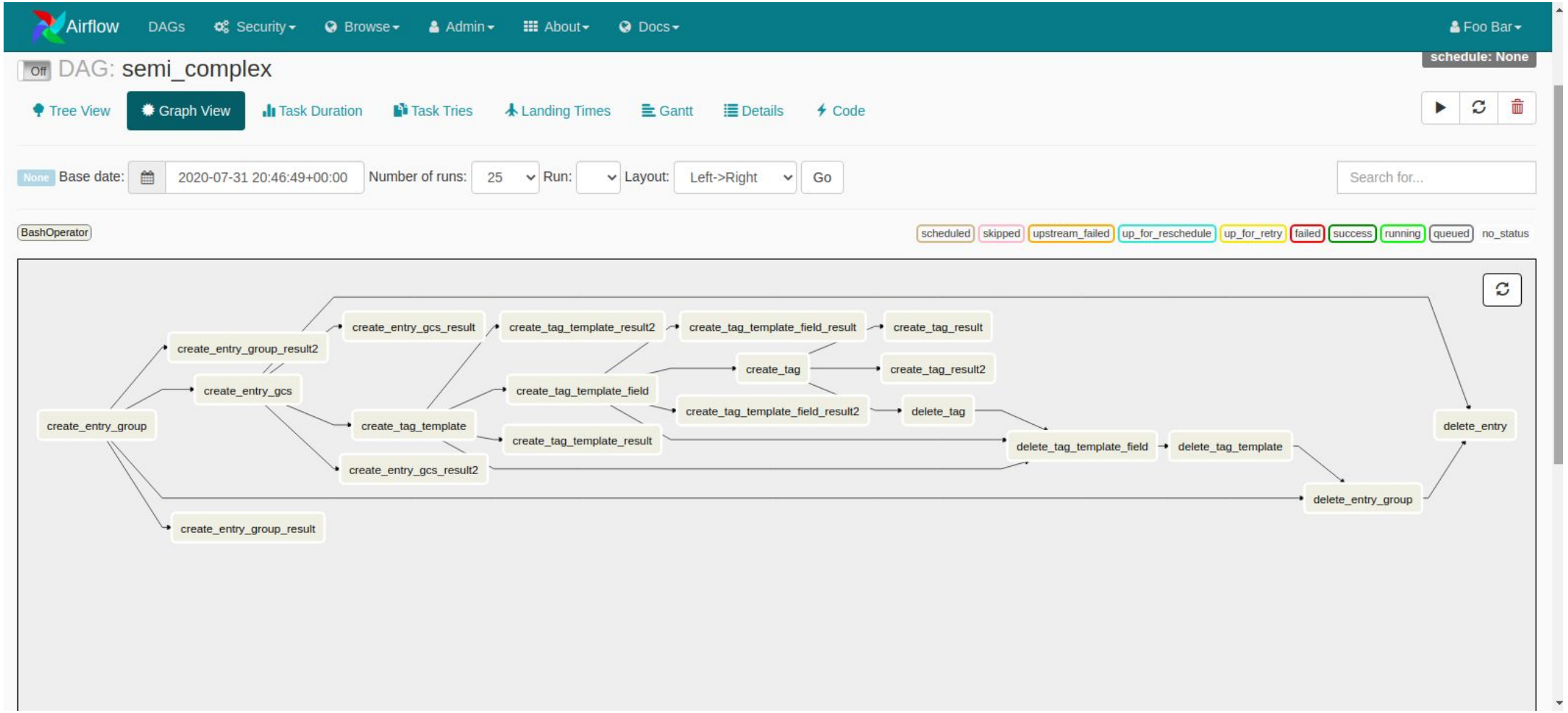
Для оркестровки потоков операций используется представление в виде **направленного ациклического графа** (DAG); собранная в граф группа операций может запускаться либо по определённому расписанию (например, ежечасно или ежедневно), либо по событию



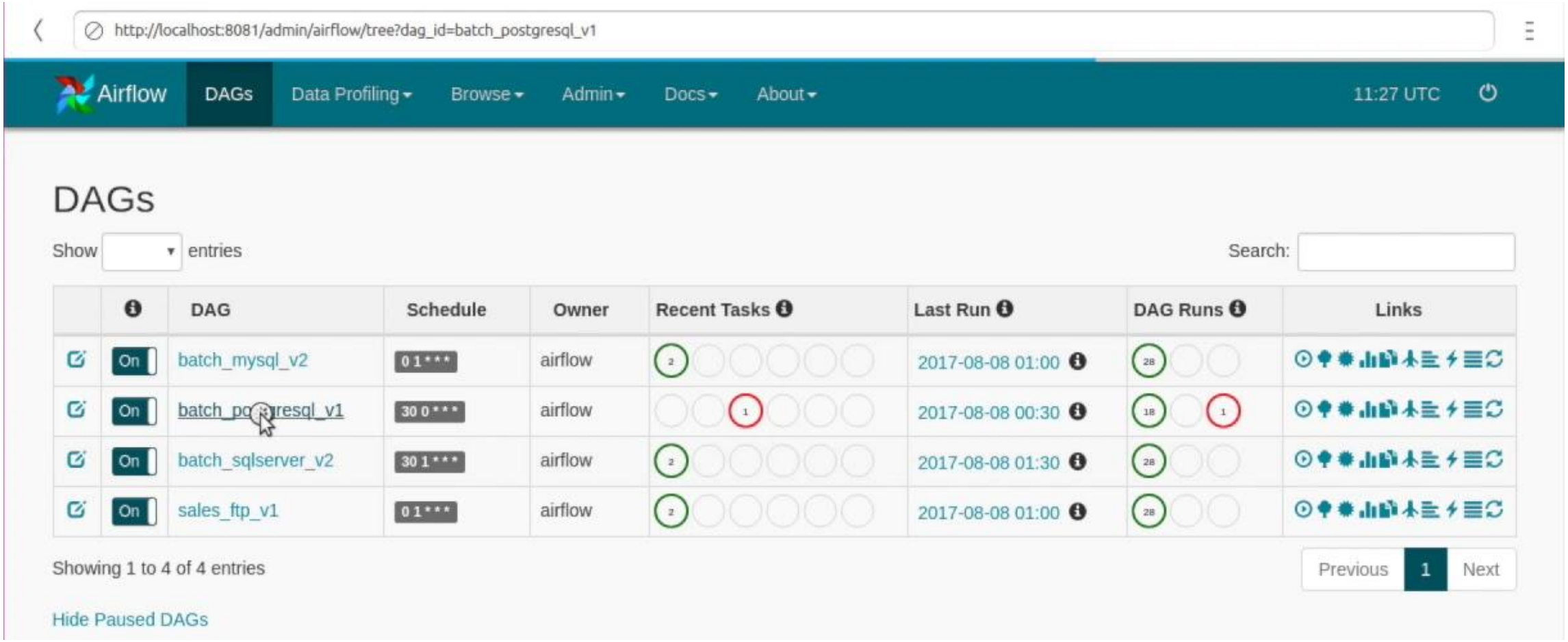
Apache Airflow



Apache Airflow

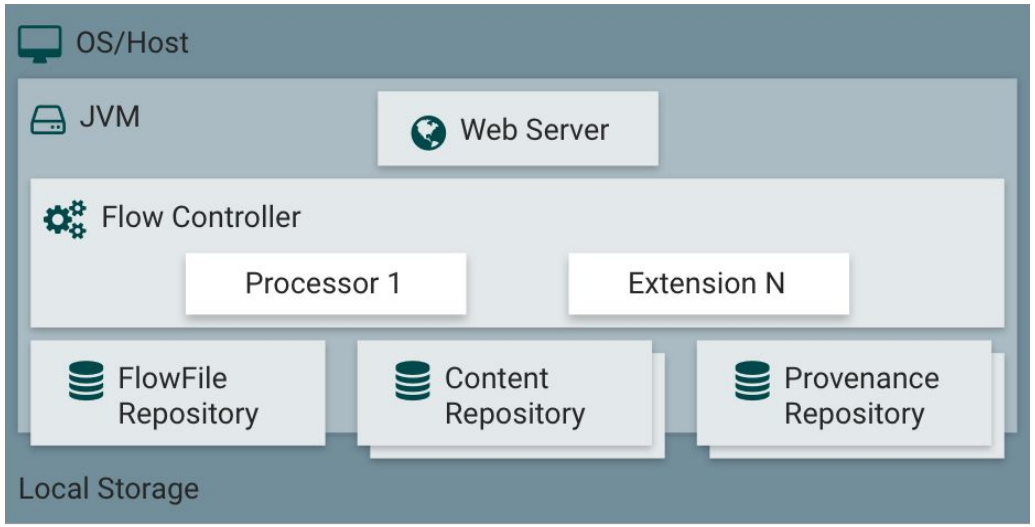


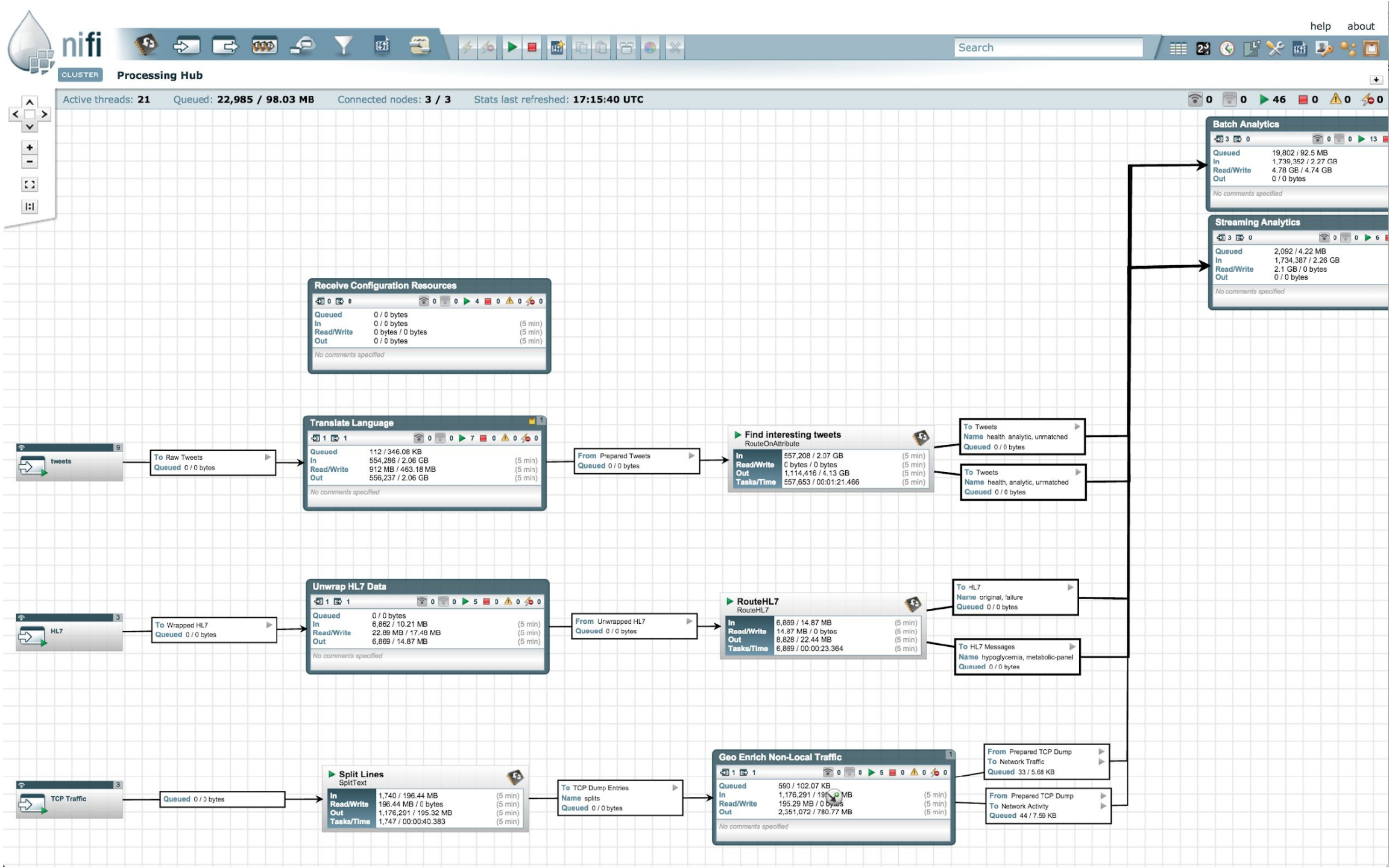
Apache Airflow



Apache Nifi

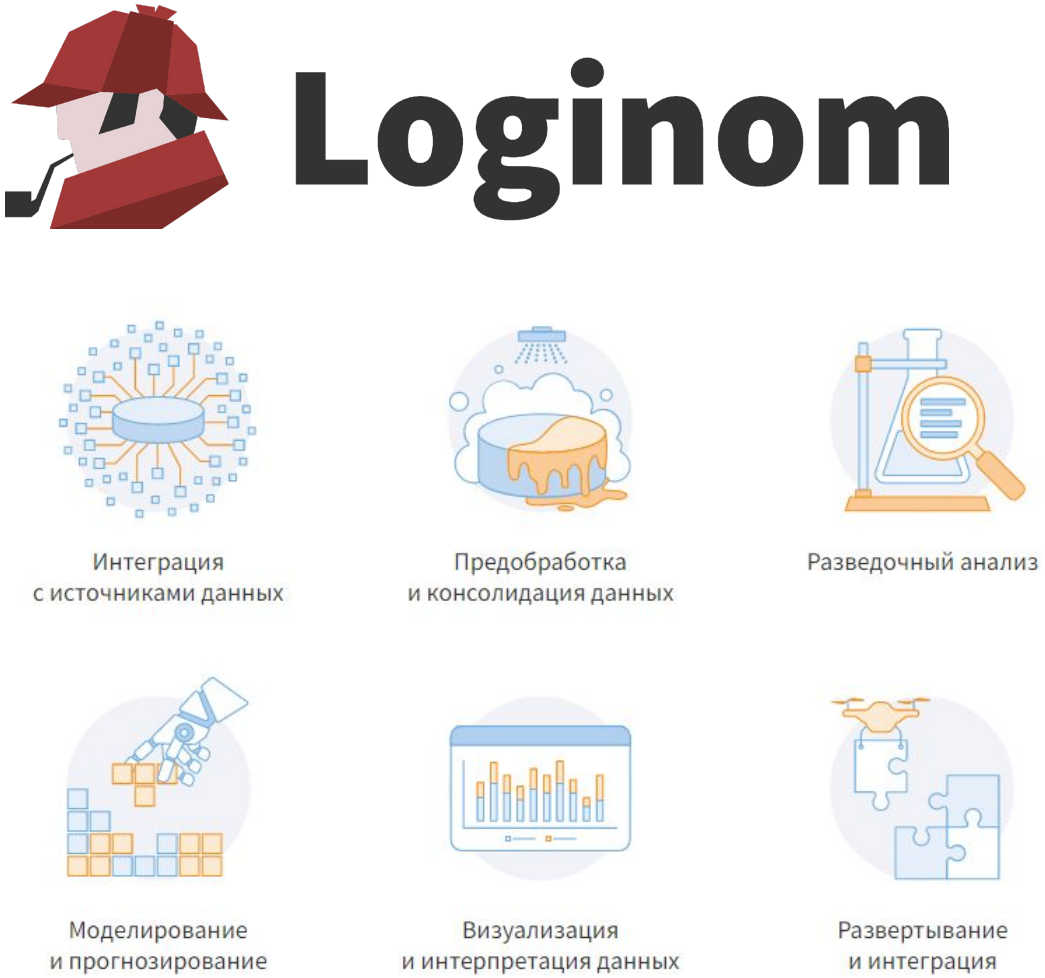
- **Apache NiFi** — это открытое программное обеспечение проекта Apache Software Foundation, предназначенное для автоматизации операций по обработке потоков данных.
- Инструмент для извлечения, преобразования, загрузки (ETL).
- Программный продукт разработан на модели программирования на основе потоков и предлагает функции, которые включают в себя возможность работы в кластерах, безопасность с использованием шифрования TLS, расширяемость и пользовательский интерфейс для визуального просмотра и изменения сценариев обработки данных.

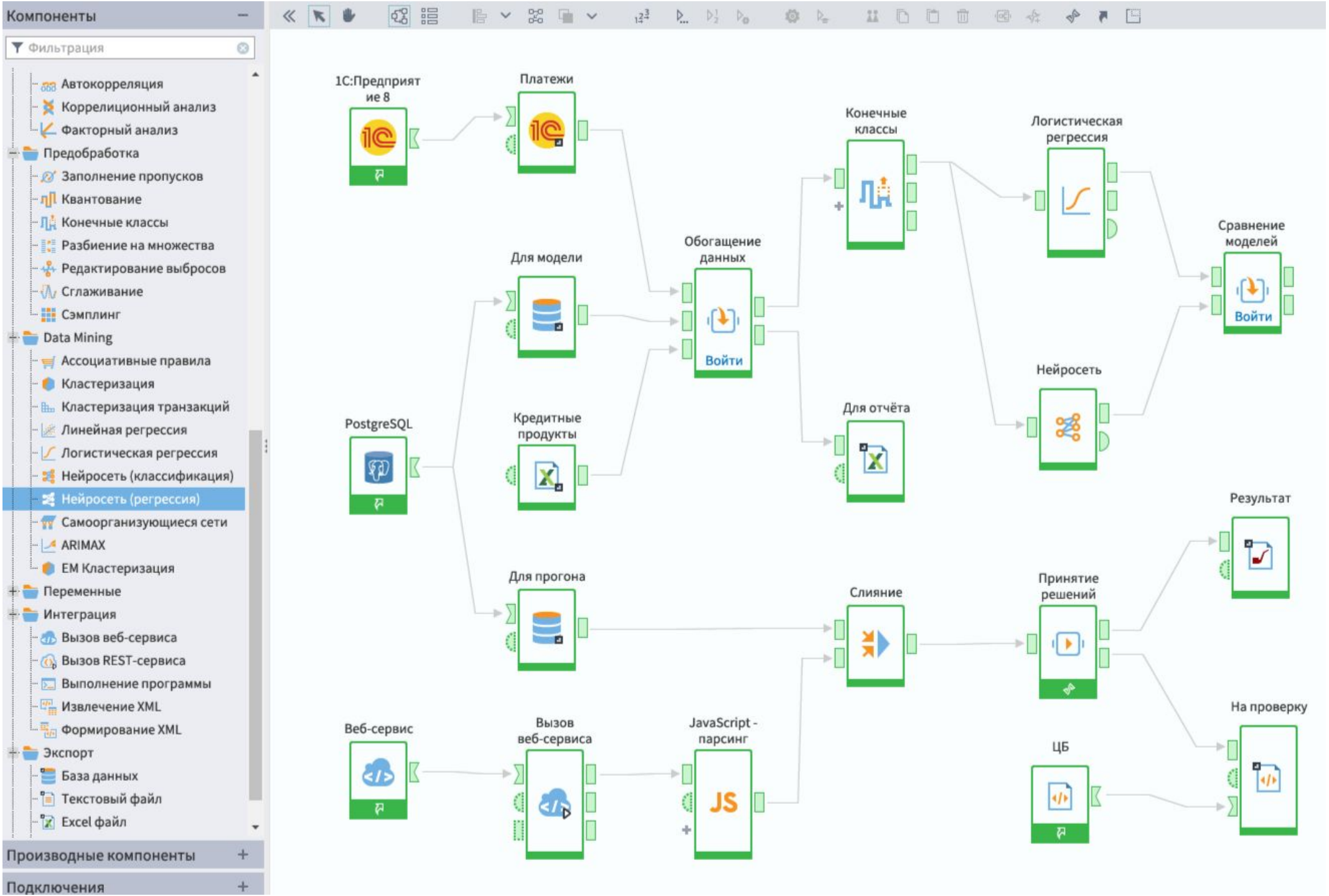



nifi
CLUSTER
Processing Hub
help
about
Search
Active threads: 21
Queued: 22,985 / 98.03 MB
Connected nodes: 3 / 3
Stats last refreshed: 17:15:40 UTC
Batch Analytics
Queued 19,802 / 92.5 MB
In 1,739,352 / 2.27 GB
Read/Write 4.78 GB / 4.74 GB
Out 0 / 0 bytes
No comments specified
Streaming Analytics
Queued 2,092 / 4.22 MB
In 1,734,387 / 2.26 GB
Read/Write 2.1 GB / 0 bytes
Out 0 / 0 bytes
No comments specified
Receive Configuration Resources
Queued 0 / 0 bytes
In 0 / 0 bytes (5 min)
Read/Write 0 bytes / 0 bytes (5 min)
Out 0 / 0 bytes (5 min)
No comments specified
tweets
To Raw Tweets
Queued 0 / 0 bytes
Translate Language
Queued 112 / 346.08 KB
In 554,286 / 2.06 GB (5 min)
Read/Write 912 MB / 463.18 MB (5 min)
Out 556,237 / 2.06 GB (5 min)
No comments specified
From Prepared Tweets
Queued 0 / 0 bytes
Find interesting tweets
RouteOnAttribute
In 557,208 / 2.07 GB (5 min)
Read/Write 0 bytes / 0 bytes (5 min)
Out 1,114,416 / 4.13 GB (5 min)
Tasks/Time 557,653 / 00:01:21.466 (5 min)
To Tweets
Name health, analytic, unmatched
Queued 0 / 0 bytes
To Tweets
Name health, analytic, unmatched
Queued 0 / 0 bytes
HL7
To Wrapped HL7
Queued 0 / 0 bytes
Unwrap HL7 Data
Queued 0 / 0 bytes
In 6,862 / 10.21 MB (5 min)
Read/Write 22.89 MB / 17.48 MB (5 min)
Out 6,869 / 14.87 MB (5 min)
No comments specified
From Unwrapped HL7
Queued 0 / 0 bytes
RouteHL7
RouteHL7
In 6,869 / 14.87 MB (5 min)
Read/Write 14.87 MB / 0 bytes (5 min)
Out 8,828 / 22.44 MB (5 min)
Tasks/Time 6,869 / 00:00:23.364 (5 min)
To HL7
Name original, failure
Queued 0 / 0 bytes
To HL7 Messages
Name hypoglycemia, metabolic-panel
Queued 0 / 0 bytes
TCP Traffic
Queued 0 / 0 bytes
Split Lines
SplitText
In 1,740 / 196.44 MB (5 min)
Read/Write 196.44 MB / 0 bytes (5 min)
Out 1,176,291 / 195.32 MB (5 min)
Tasks/Time 1,747 / 00:00:40.383 (5 min)
To TCP Dump Entries
Name splits
Queued 0 / 0 bytes
Geo Enrich Non-Local Traffic
Queued 590 / 102.07 KB
In 1,176,291 / 195.20 MB (5 min)
Read/Write 195.29 MB / 0 bytes (5 min)
Out 2,351,072 / 780.77 MB (5 min)
No comments specified
From Prepared TCP Dump
To Network Traffic
Queued 33 / 5.68 KB
From Prepared TCP Dump
To Network Activity
Queued 44 / 7.59 KB

Loginom

- **Loginom** — аналитическая платформа, позволяющая в единой среде выполнить все этапы бизнес-анализа от консолидации данных и построения моделей до визуализации и интеграции в бизнес-процесс.
- Инструмент для извлечения, преобразования, загрузки (ETL).
- Для решения задач анализа **Loginom** позволяет импортировать данные из различных источников и применять к ним необходимые алгоритмы обработки. Результаты можно просмотреть в самой системе или экспортировать в сторонние приемники данных.





МИРЭА
Российский технологический университет

Источники информации

1. Принципы построения систем потоковой аналитики. Блог компании OTUS. / https://habr.com/ru/company/otus/blog/477834/
2. Обзор состояния области потоковой обработки данных. Р.С. Самарев / https://www.ispras.ru/proceedings/docs/2017/29/1/isp_29_2017_1_231.pdf?ysclid=ld056gcdyv572223822
3. Apache Airflow Platform / https://airflow.apache.org
4. Apache NiFi Platform / https://nifi.apache.org/
5. Что такое озеро данных? Avijit Prasad | Консультант по облачным решениям / https://learn.microsoft.com/ru-ru/azure/architecture/data-guide/scenarios/data-lake
6. Аналитическая Low-code платформа Loginom / https://loginom.ru